SHARP

POCKET COMPUTER

MODEL PC-1350

INSTRUCTION MANUAL

MODEL PC-1350 AND PERIPHERALS
LIMITED WARRANTY

Sharp Electronics Corporation warrants each of these products to the original purchaser to be free from defective materials and workmanship. Under this warranty the product will be repaired or replaced, at our option, without charge for parts or labor, with the exception of supplies, such as batteries, ribbons, inked rollers, etc., when returned to a SHARP FACTORY SERVICE CENTER listed in the instruction booklet supplied with your product.

This warranty does not apply to cassette tapes, software programs or appearance items nor to any product whose exterior has been damaged or defaced, nor to any product subjected to misuse, abnormal service or handling, nor to any product altered or repaired by other than a SHARP FACTORY SERVICE CENTER. This warranty does not apply to any product purchased outside the United States, its territories or possessions.

The period of the warranty shall be ninety (90) days on parts and labor from the date of the original purchase.

This warranty entitles the original purchaser to have the warrantied parts and labor rendered at no cost for the period of the warranty described above when the unit is carried or shipped prepaid to a SHARP FACTORY SERVICE CENTER together with proof of purchase.

THIS SHALL BE THE EXCLUSIVE WRITTEN WARRANTY OF THE ORIGINAL PURCHASER AND NEITHER THIS WARRANTY NOR ANY OTHER WARRANTY, EXPRESSED OR IMPLIED SHALL EXTEND BEYOND THE PERIOD OF TIME LISTED ABOVE. IN NO EVENT SHALL SHARP BE LIABLE FOR CONSEQUENTIAL ECONOMIC DAMAGE OR CONSEQUENTIAL DAMAGE TO PROPERTY. SOME STATES DO NOT ALLOW A LIMITATION ON HOW LONG AN IMPLIED WARRANTY LASTS OR AN EXCLUSION OF CONSEQUENTIAL DAMAGE, SO THE ABOVE LIMITATION AND EXCLUSION MAY NOT APPLY TO YOU. IN ADDITION, THIS WARRANTY GIVES SPECIFIC LEGAL RIGHTS AND YOU MAY HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

FOR YOUR RECORDS . . . .

For your assistance in reporting this electronic calculator in case of loss or theft, please record below the model number and serial number which are located on the bottom of the unit.
Please retain this information.

Model Number_____ Serial Number_____

Date of Purchase _____ Place of Purchase _____

TABLE OF CONTENTS

Page

INTRODUCTORY NOTE 4

CHAPTER 1 HOW TO USE THIS MANUAL 5

CHAPTER 2 INTRODUCTION TO THE PC-1350 7
- Description of Keys 8
- Description of Display 12
- ALL RESET Button 14
- Cell Replacement 15
- Installing the Cells 15

CHAPTER 3 USING THE PC-1350 AS A CALCULATOR 18
- Start Up 18
- Shut Down 18
- Auto Off 18
- Some Helpful Hints 19
- Simple Calculations 20
- Recalling Entries 21
- Errors 25
- Serial Calculations 25
- Negative Numbers 27
- Compound Calculations and Parentheses 27
- Using Variables in Calculations 28
- Chained Calculations 29
- Scientific Notation 30
- Limits 31
- Last Answer Feature 31
- Length of Formula 33
- Scientific Calculations 33
- Priority in Manual Calculation 35
- Printing for Manual Calculation 36

CHAPTER 4 CONCEPTS AND TERMS OF BASIC 38
- String Constants 38
- Hexadecimal Numbers 38
- Variables 38
- Fixed Variables 40
- Simple Variables 41
- Array Variables 41
- Variables in the Form of A( ) 45
- Expressions 46
- Numeric Operators 47

String Expressions 47
Relational Expressions 47
Logical Expressions 48
Parentheses and Operator Precedence 50
RUN Mode 51
Functions 51

CHAPTER 5 PROGRAMMING THE PC-1350 52
Programs 52
BASIC Statements 52
Line Numbers 52
BASIC Verbs 53
BASIC Commands 53
Modes 54
Beginning to Program on the PC-1350 54
Example 1—Entering and Running a Program 54
Example 2—Editing a Program 55
Example 3—Using Variables in Programming 58
Example 4—More Complex Programming 61
Storing Programs in the PC-1350's Memory 62
Graphic Functions 63
Serial I/O Functions 67

CHAPTER 6 SHORTCUTS 69
The DEF Key and Labelled Programs 69
RESERVE Mode 70
Templates 72

CHAPTER 7 USING THE CE-126P PRINTER/CASSETTE INTERFACE 73
Using the Printer 73
Using the Cassette Interface 75
Tape Notes 80

CHAPTER 8 USING THE RAM CARD (OPTION) 81
Mounting the RAM Card 81
Removing the RAM Card 83
Using the RAM Card 84
Saving Data within the RAM Card 85
Precautions When Using the RAM Card 86

CHAPTER 9 BASIC REFERENCE 88
Commands 92
Verbs 106
Functions 156
Pseudovariables 156
Numeric Functions 160

String Functions 164
Graphic Related Commands 166
Serial I/O Related Commands 179
Text Function Related Commands 194

CHAPTER 10 TROUBLESHOOTING 197
Machine Operation 197
BASIC Debugging 198

CHAPTER 11 MAINTENANCE OF THE PC-1350 202

APPENDICES 203
Appendix A: Error Messages 203
Appendix B: Character Code Chart 206
Appendix C: Formatting Output 209
Appendix D: Expression Evaluation and Operator Priority 213
Appendix E: Key Functions in BASIC 215
Appendix F: Signals Used in the Serial I/O Terminal 219
Appendix G: Specifications 221
Appendix H: Using Program Developed for the PC-1210 SERIES, PC-1245 SERIES, PC-1260 SERIES or PC-1401 223

PROGRAM EXAMPLES 229

INTRODUCTORY NOTE

Welcome to the world of SHARP owners!

Few industries in the world today can match the rapid growth and technological advances being made in the field of personal computing. Computers which just a short time ago would have filled a huge room, required a Ph.D. to program, and cost thousands of dollars, now fit in the palm of your hand, are easily programmed, and cost so little that they are within the reach of nearly everyone.

Your new SHARP PC-1350 was designed to bring you all of the latest state of the art features of this computing revolution and it incorporates many advanced capabilities:

* MEMORY SAFE GUARD—the PC-1350 remembers stored programs and variables even when you turn it off.
* Battery powered operation for true portability.
* AUTO POWER OFF function which conserves the batteries by turning the power off if no activity takes place within a specified time limit.
* An expanded version of BASIC which provides formatted output, two-dimensional arrays, variable length strings, program chaining and many other advanced features.
* An optional printer/cassette interface (Model CE-126P) with the printer, you can have "hard-copies" of programs and data. The cassette interface lets you connect a cassette recorder to store programs and data to cassette recorder.

Congratulations on entering an exciting and enjoyable new world. We are sure that you will find this purchase one of the wisest you have ever made. The SHARP PC-1350 is a powerful tool, designed to meet your specific mathematical, scientific, engineering, business and personal computing needs. With the SHARP PC-1350 you can begin NOW providing the solutions you'll need tomorrow!

CHAPTER 1
HOW TO USE THIS MANUAL

This manual is designed to introduce you to the capabilities and features of your PC-1350 and to serve as a valuable reference tool. Whether you are a "first time user" or an "old hand" with computers, you should acquaint yourself with the PC-1350 by reading and working through Chapters 2 through 6.

* Chapter 2 describes the physical features of the PC-1350.
* Chapter 3 demonstrates the use of the PC-1350 as a scientific calculator.
* Chapter 4 defines some terms and concepts which are essential for BASIC programming, and tells you about the special considerations of these concepts on the PC-1350.
* Chapter 5 introduces you to BASIC programming on the PC-1350, showing you how to enter, correct, and run programs.
* Chapter 6 discusses some shortcuts that make using your new computer easier and more enjoyable.

Experienced BASIC programmers may then read through Chapter 9 to learn the specific features of BASIC as implemented on the PC-1350. Since every dialect of BASIC is somewhat different, read through this material at least once before starting serious programming.

Chapter 9 is a reference section covering all the verbs, commands, and functions of BASIC arranged in convenient alphabetical groupings.

If you have never programmed in BASIC before, we suggest that you buy a separate book on beginning BASIC programming or attend a BASIC class, before trying to work through these chapters. This manual is not intended to teach you how to program.

The remainder of the manual consists of:

* Chapter 7 —Basic information on the optional CE-126P Printer/Cassette Interface.
* Chapter 8 —Description for the use of the optional CE-201M/202M RAM Card.
* Chapter 10—A troubleshooting guide to help you solve some operating and programming problems.
* Chapter 11—The care and maintenance of your new computer.

Detailed Appendices, at the end of the manual, provide you with useful charts, and special discussions concerning the use and operation of the PC-1350.

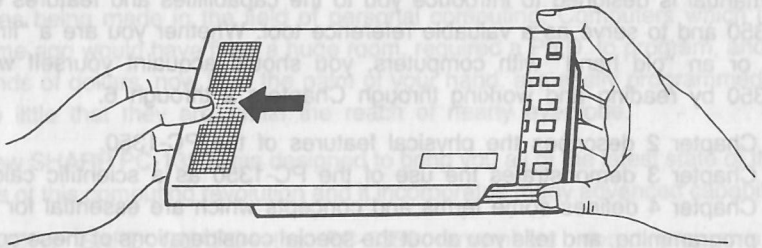
Using the Hard Cover

When the computer is not being used, mounted the hard cover on the operation panel of the computer.

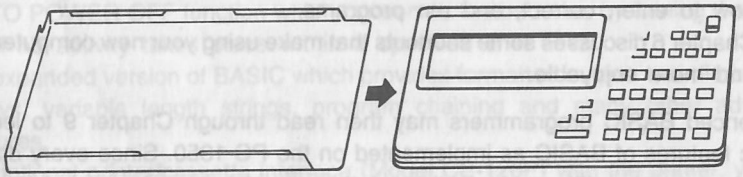
● When the computer is to be used.

To remove the hard cover from the computer, remove it as shown in figure below.

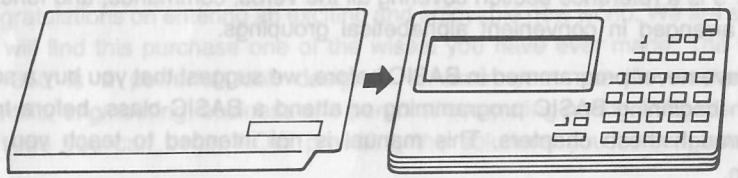
Step ①



Step ②



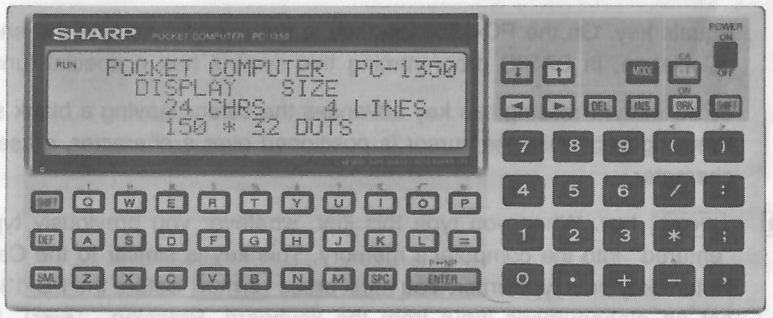
● When the computer is not used.



CHAPTER 2
INTRODUCTION TO THE PC-1350

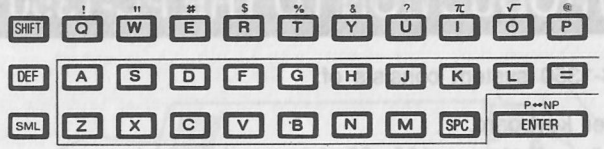
The SHARP PC-1350 system consists of:

* 62-character keyboard.
* 24-digit 4 line display or 150×32 dots graphic display.
* 8-bit CMOS processor.
* 40KB ROM.
* 5KB RAM. (standard)
* Optional CE-126P. (Printer/Cassette Interface)



To familiarize you with the placement and functions of parts of the PC-1350 keyboard, we will now study each section of the keyboard. For now just locate the keys and read the description of each. In Chapter 3 we will begin using your new machine.

DESCRIPTION OF KEYS



[A] ~ [Z] The Sharp PC-1350 has the same 26-letter arrangement found on most typewriters. The letters are normally upper case (the opposite of most typewriters), but this is convenient because the PC-1350 only recognizes statements and commands in upper case letters. To enter the lower case letters, set the PC-1350 to the LOWER CASE mode by pressing the [SML] key. (See the next page.)

[=] Equals key. On the PC-1350 this key is not used to indicate the end of a calculation; in BASIC programming this symbol has a special function.

[SPC] SPaCe key. Pressing this key advances the cursor leaving a blank space. Pressing [SPC] while the cursor is positioned over a character, erases that character.

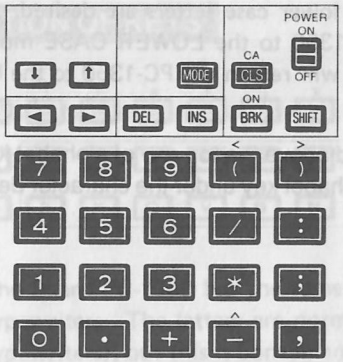
[P↔NP ENTER] [ENTER] key. When you type this key, whatever you previously typed is "entered" into the computer's memory. This key is similar to the Carriage Return key on a typewriter. You must press [ENTER] before the PC-1350 will act on alphanumeric input from the keyboard. Pressing [SHIFT] before pressing this key will cause the PC-1350 to switch on and off the printing of calculations on the optional printer.

[DEF] [DEF] key. This is a special key used to execute BASIC programs.

[SHIFT] The [SHIFT] key has two important uses. First, if the [SHIFT] key is pressed before a key with a secondary function printed above it, the secondary function will be called in (e.g., pressing the [SHIFT] key before pressing the [CLS] key will call in the Clear All (CA) function). Second, if the [SHIFT] key is pressed before pressing the [MODE] key, the REVERSE mode will be selected. Ordinarily, pressing the [MODE] key will alternately select the RUN mode or the PROGRAM mode. Using the [SHIFT] key makes a third mode possible. (See Mode key explanation in this section for more detail.)

[SML] The [SML] key is used when lower case letters are desired. Pressed once, this key will change the PC-1350 to the LOWER CASE mode ("SML" is displayed). Pressed again, it will return the PC-1350 to the UPPER CASE mode ("SML" disappears).

! " # $ % & ? π √ @ These symbols are found above the top row of alphabet keys. Pressing [SHIFT] and then the alphabet key under the character desired displays these symbols.



POWER ON / OFF — Use this power slide switch to turn the PC-1350 ON and OFF.

SHIFT — Shift key. See page 8.

CA CLS — The red Clear CLS key allows you to clear the display screen. Pessing SHIFT before CLS will activate the Clear All secondary function. This will reset the computer, clearing the display and the memory stack.
NOTE: The Clear All function will not erase all programs, reserve memories, and all variables. To erase all programs, type in **NEW** and press ENTER while in the PROGRAM mode. To erase all reserve memories, type in **NEW** and press ENTER while in the RESERVE mode. To clear all variables by setting them to zero (or null), type in **CLEAR** and press ENTER .

MODE — When you turn the PC-1350 on, check the display for the operational mode it is in (RUN, PROGRAM, or RESERVE). Press MODE to change the selection from RUN to PROGRAM or from PROGRAM to RUN. Press SHIFT and then press MODE to select the RESERVE mode. To leave the RESERVE mode, press MODE . All manual calculations (using the computer as a calculator) are done in the RUN mode, and all programs are run in the RUN mode. All programs are written and edited in the PROGRAM mode. The RESERVE mode is used for storing frequently used functions on single keys and for storing menus to help identify these functions quickly.

ON BRK — BREAK key. The BRK key temporarily interrupts a program which is being executed. Pressing this key after an AUTO OFF turns the computer back on.

INS — Insert key. Pressing this key makes a space directly before the character the cursor is "on top of." You can then INSert new characters into this space.

DEL — Delete key. Pressing this key will DELete whatever character the cursor is "on top of."

↓ — Down Arrow key. Press this key to display the next line.

↑ — Up Arrow key. Press this key to display the previous line.

◄ — Backspace key. This key allows you to move the cursor to the left without erasing previously typed characters.

► — Forward key. This key allows you to move the cursor to the right without erasing previously typed characters.

0 ~ 9 . — Number keys. The layout of these keys is similar to that found on the standard calculator.

/ — Division key. Press this key to include the division operator in calculations.

* — Multiplication key. Press this key to include the multiplication operator in calculations.

^ − — Subtraction key. Press this key to include the subtraction operator in calculations.
Pressing SHIFT and then this key will display the "power" symbol, indicating that a number is to be raised to a specific power.

+ — Addition key. Press this key to include the addition operator in calculations.

< (— Left parenthesis key.
Pressing SHIFT and then this key displays the "less than" character.

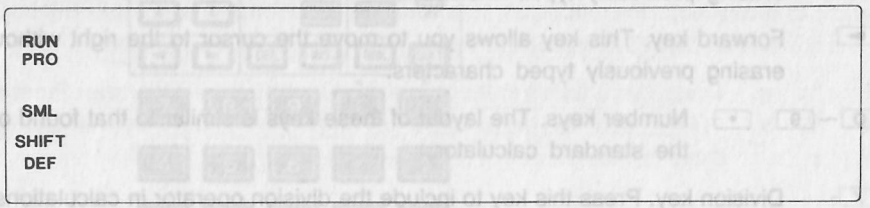
>) — Right parenthesis key.
Pressing SHIFT and then this key displays the "greater than" character.

: — Colon key.

; — Semicolon key.

, — Comma key.

DESCRIPTION OF DISPLAY



The PC-1350 has a 150×32 programmable, dot-matrix liquid crystal display. This display shows up to 24 characters per a line and total 4 lines. Each character occupies a 5×7 dot matrix.
For graphic purposes, the entire display may be utilized as a 150×32 dot matrix. Individual dots within any of 150 columns may be energized to create graphics, figures, or special symbols.

The display consists of:

The prompt. This symbol appears when the computer is awaiting input. As you type, the prompt disappears and is replaced by the cursor.

The cursor. This symbol (the underline) tells you the location of the next character to be typed in. As you begin typing, the cursor replaces the prompt. The cursor is also used to position the computer over certain characters when using the INSert and DELete functions.

RUN RUN indicator. This indicator tells you the operational mode of the PC-1350 is in the RUN mode.

PRO PROgram indicator. This indicator tells you the operational mode of the PC-1350 is in the programming mode.

Note: If neither RUN nor PRO indicator can be found on the display, the PC-1350 is in the reserve (RSV) mode.

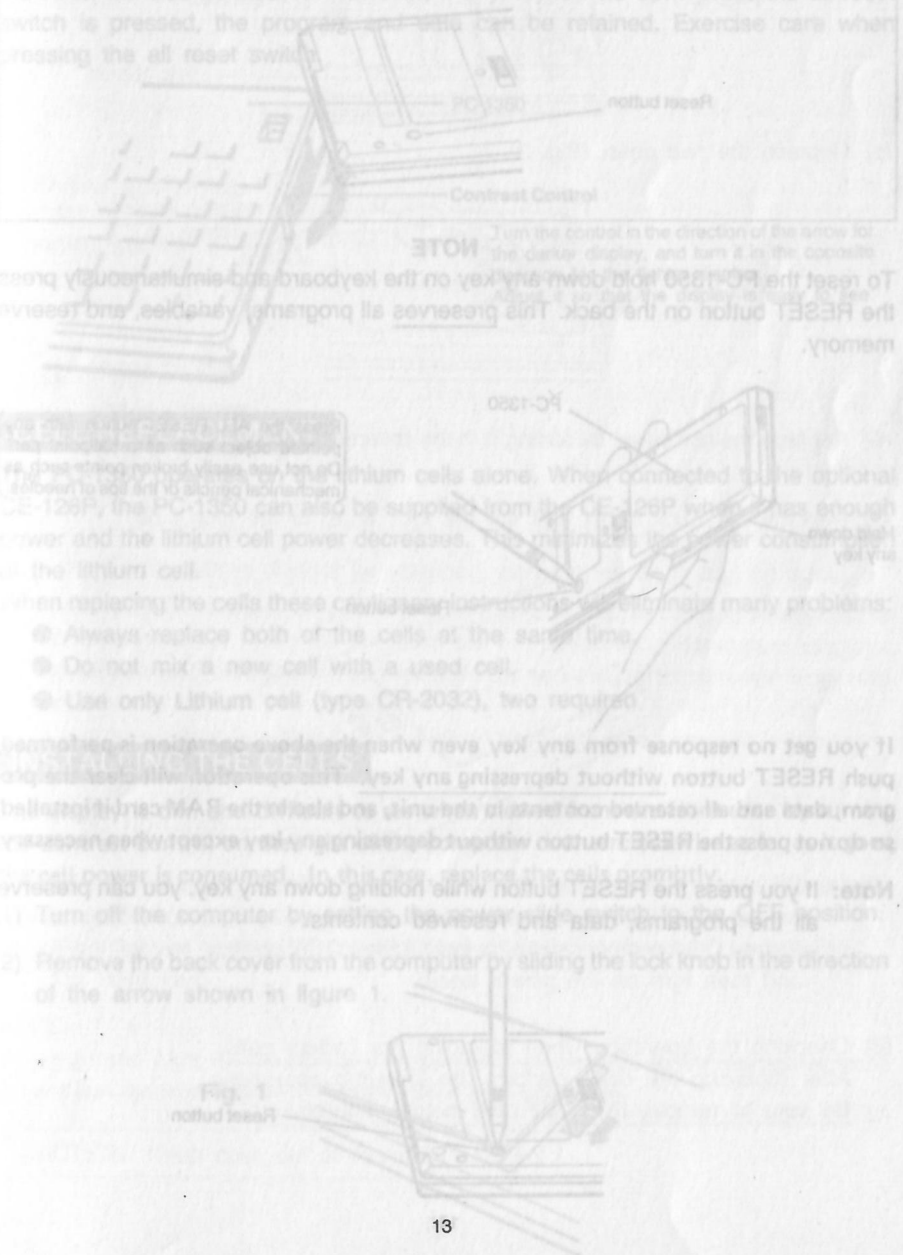
SML Lower case mode indicator. "SML" is displayed when [SML] is pressed. When this indicator is shown, the alphabet keys are entered as lower case characters. When [SML] is pressed while the "SML" indicator is shown, the indicator disappears and the computer returns to the upper case character mode.

SHIFT Shift Key Indicator. This indicator displays when the [SHIFT] key has been depressed. Remember, the [SHIFT] key must be released before depressing any other key.

DEF Definable Mode Indicator. This symbol lights up when you press the [DEF] key.

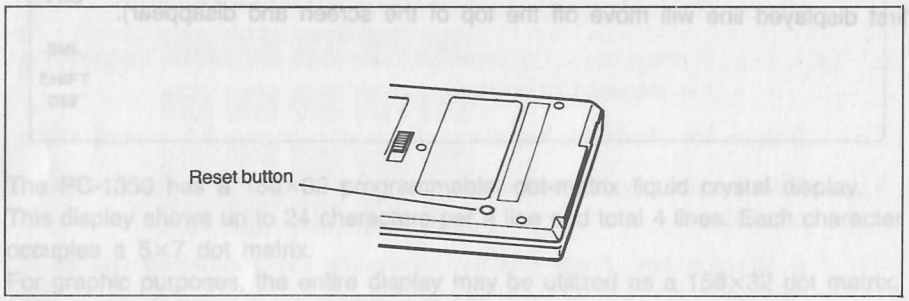
For Displays Exceeding 4 Lines

The display unit of the PC-1350 consists of 4 lines (24 characters per line). Key inputs or calculated results are displayed from the top line of the display. If the characters to be displayed exceed 4 lines, the displayed contents will be moved up by 1 line (the first displayed line will move off the top of the screen and disappear).



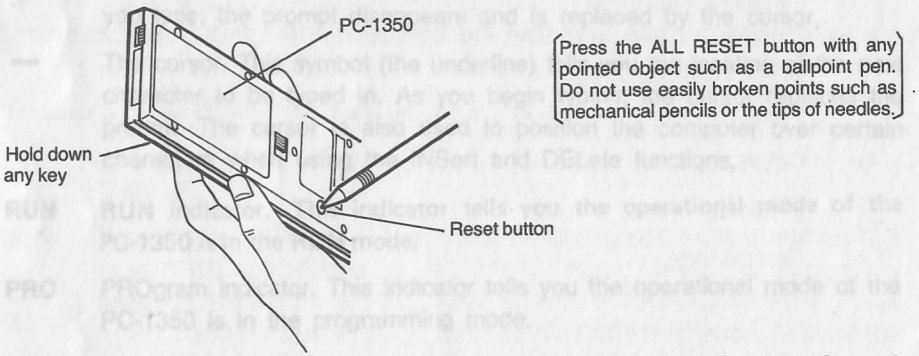
All RESET Button

ALL RESET: Reset button. This button is used to reset the computer when CLear ([CLS]) or CA is not sufficient to correct the problem.



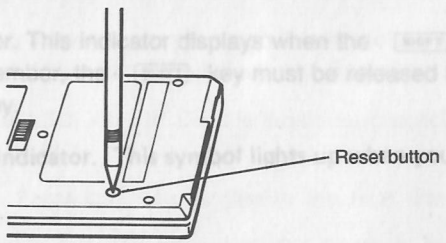
NOTE

To reset the PC-1350 hold down any key on the keyboard and simultaneously press the RESET button on the back. This preserves all programs, variables, and reserve memory.


PC-1350
Press the ALL RESET button with any pointed object such as a ballpoint pen. Do not use easily broken points such as mechanical pencils or the tips of needles.
Hold down any key
Reset button

If you get no response from any key even when the above operation is performed, push RESET button without depressing any key. This operation will clear the program, data and all reserved contents in the unit, and also in the RAM card if installed, so do not press the RESET button without depressing any key except when necessary.

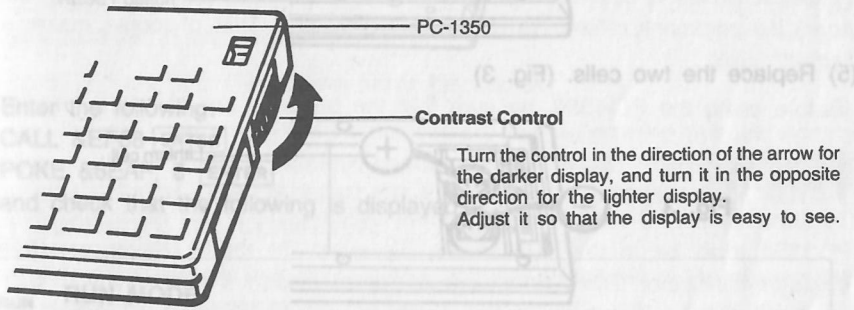
Note: If you press the RESET button while holding down any key, you can preserve all the programs, data and reserved contents.



If the unit still does not operate normaly, remove the lithium cells. After waiting about 10 seconds, insert the cells and press the all reset switch.

Function of the all reset switch

If only the all reset switch is pressed, the program and data will be completely cleared. If, at this time, the PC-1350 is mounted with the RAM card, the program and data on the RAM will also be cleared. However, if any key is held down while the all reset switch is pressed, the program and data can be retained. Exercise care when pressing the all reset switch.


PC-1350
Contrast Control
Turn the control in the direction of the arrow for the darker display, and turn it in the opposite direction for the lighter display.
Adjust it so that the display is easy to see.

CELL REPLACEMENT

The PC-1350 operates on the lithium cells alone. When connected to the optional CE-126P, the PC-1350 can also be supplied from the CE-126P when it has enough power and the lithium cell power decreases. This minimizes the power consumption of the lithium cell.

When replacing the cells these cautionary instructions will eliminate many problems:

- Always replace both of the cells at the same time.
- Do not mix a new cell with a used cell.
- **Use only Lithium cell (type CR-2032),** two required

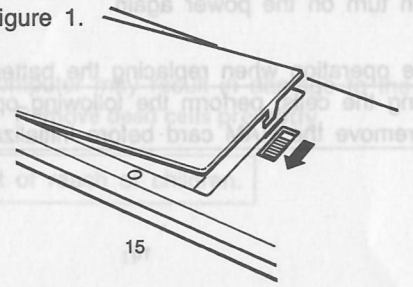
INSTALLING THE CELLS

The display is dim and difficult to see when viewed from the front even by turning the contrast control on the right of the computer counterclockwise as far as it goes, the cell power is consumed. In this case, replace the cells promptly.

(1) Turn off the computer by setting the power slide switch to the OFF position.

(2) Remove the back cover from the computer by sliding the lock knob in the direction of the arrow shown in figure 1.

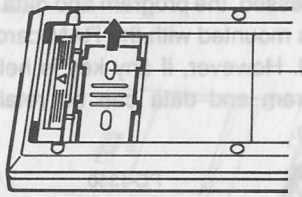
Fig. 1



(3) If there is a optional RAM card in the card slot, remove it in accordance with the procedure on page 83.

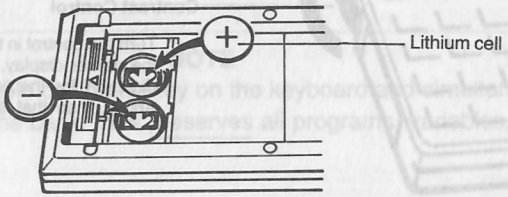
(4) Remove the cell cover by sliding it in the direction of the arrow shown in Figure.

Fig. 2



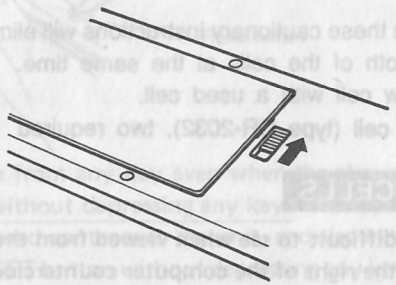
(5) Replace the two cells. (Fig. 3)

Fig. 3



(6) Replace the cell cover by sliding it in the reverse direction of the arrow shown in figure 2.

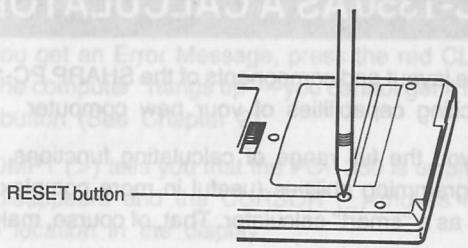
(7) Replace the back cover and set the lock knob to the lock side.



Notes: Be sure to set the lock to the LOCK position. The PC-1350 will not operate if not set to the LOCK position. If the power is turned on without setting the lock to the LOCK position, slide the lock to the LOCK position, turn off the power, and then turn on the power again.

(8) Checking the operation when replacing the battery cells
After replacing the cells, perform the following operations.
Be sure to remove the RAM card before initialization.

① Turn on the computer by setting the power slide switch to the ON position and press the RESET button to clear the computer.



② Enter the following:
CALL &EF88 [ENTER]
POKE &6EAF, 0 [ENTER]
and check that the following is displayed:

```
RUN  RUN MODE
     CALL &EF88
     POKE &6EAF, 0
     >
```

③ If the display above cannot be obtained, perform steps ① and ② again.
CALL &EF88 initializes the serial I/O function. POKE &6EAF, 0 initializes the display function. If these initializations are not performed meaningless displays may appear or the equipment connected to the serial I/O terminal may not operate properly.
If a program and data have been stored on the RAM card, remount it after performing the initialization.
If the PC-1350 is initialized while the RAM card is mounted, the RAM card will also be initialized (program and data cleared).

(9) When you removed the RAM card from the computer in above step (3), replace the card into the computer. For installing the card, refer to page 81.
Note that you must do the check of correct cell replacement prior to installing the RAM card.

NOTE:
Keeping dead cells in the computer may result in damage to the computer from solvent leakage from the cell. Remove dead cells promptly.

CAUTION: Keep cells out of reach of children.

CHAPTER 3
USING THE PC-1350 AS A CALCULATOR

Now that you are familiar with the layout and components of the SHARP PC-1350, we will begin investigating the exciting capabilities of your new computer.

Because the PC-1350 allows you the full range of calculating functions, plus the increased power of BASIC programming abilities (useful in more complex calculations), it is commonly referred to as a "smart" calculator. That, of course, makes you a "smart" user!

(Before using the PC-1350, be sure that the batteries are correctly installed.)

Start Up

To turn ON the PC-1350, slide the power switch up. For use as a calculator, the PC-1350 must be in the RUN mode. To select the mode setting, use [MODE] to display the incicator "RUN" on the left side of the display. When the machine is ON, the prompt (>) will appear on the display.

Shut Down

To turn OFF the PC-1350, slide the power switch to the OFF position.

When you turn OFF the machine, you clear (erase) the display. However, the PC-1350 does remember all programs and reserve keys settings which were in use when the computer was turned OFF. All of these settings are still in effect when the machine is turned back ON.
When the CLOAD command is executed, stop the execution by pressing the [BRK] key and slide the power switch to the OFF position.

Auto Off

In order to conserve on battery wear, the PC-1350 automatically powers down when no keys have been pressed for about 11 minutes. (Note: The PC-1350 will not AUTO OFF while you are executing a program.)

To restart the computer after an AUTO OFF, press the [ON BRK] key. All settings will be exactly as they were when the AUTO OFF occured.

Some Helpful Hints

Until you get used to your new machine, you are bound to make mistakes while entering data. Later we will discuss some simple ways to correct these mistakes. For now, if you get an Error Message, press the red CLear ([CLS]) key and retype the entry. If the computer "hangs up"—you cannot get it to respond at all—press the ALL RESET button (See Chapter 2).

The PROMPT (>) tells you that the PC-1350 is awaiting input. As you enter data the prompt disappears and the CURSOR (_) moves to the right indicating the next available location in the display.

The right [►] and left [◄] arrows move the cursor.

Pressing [ENTER] informs the PC-1350 that you are finished entering data and signals the computer to perform the indicated operations. **YOU MUST PRESS [ENTER] AT THE END OF EACH LINE OF INPUT OR YOUR CALCULATIONS WILL NOT BE ACTED UPON BY THE COMPUTER.**

When performing numeric calculations input appears on the left of the display: the results appear on the right of the display.

When using the [SHIFT] key in conjunction with another key (to access square root for example) press [SHIFT] , release the [SHIFT] , then press the other key. [SHIFT] is active for only one key at a time.

Do not use dollar signs or commas when entering calculations into the PC-1350 These characters have special meaning in the BASIC programming language.

In this manual we use the Ø to indicate zero, so that you can distinguish between the number (Ø) and the letter (O).

To help get you started entering data correctly, we will show each keystroke necessary to type in the example calculations. When [SHIFT] is used we will represent the desired character in the following keystroke. For example pressing [SHIFT] and [! Q] will produce the ! character. These keystrokes are written [SHIFT] [!] .

Be sure to enter CLear after each calculation (unless you are performing serial calculations). CLear erases the display and resets the error condition. It does not erase anything stored in the computer's memory.

Simple Calculations

The PC-1350 performs calculations with ten-digit precision. Turn ON your computer and set it in the RUN mode. Now try these simple arithmetic examples.

Input	Display
5 Ø + 5 Ø ENTER	50+50 100.
1 Ø Ø − 5 Ø ENTER	50+50 100. 100−50 50.
6 Ø * 1 Ø ENTER	100−50 50. 60*10 600.

The display unit of the PC-1350 consists of 4 llines (24 characters per line). Key inputs or calculated results are displayed from the top line of the display. If the characters to be displayed exceed 4 lines, the displayed ocntents will be moved up by 1 line (the first displayed line will move off the top of the screen and disappear).

Input	Display
3 Ø Ø / 5 ENTER	60*10 600. 300/5 60.
1 Ø SHIFT ^ 2 ENTER	300/5 60. 10^2 100.

2 * SHIFT π ENTER	10^2 100. 2*π 6.283185307
SHIFT √ 6 4 ENTER	2*π 6.283185307 √ 64 8.
4 E 3 ENTER	√ 64 8. 4E3 4000.

Recalling Entries

Even after the PC-1350 has displayed the results of your calculation, you can edit your last entry. To edit, use the left ◄ and right ► arrows.

The left arrow ◄ is used to position the cursor after the last character.

The right arrow ► is used to position the cursor "on top of" the first character.

Remember that the left and right arrows are also used to position the cursor along a line. The right and left arrows are very helpful in editing (or modifying) entries without having to retype the entire expression.

You will become familiar with the use of the right and left arrows in the following examples. Now, take the role of the manager and perform the calculations as we discuss them.

As the head of personnel in a large marketing division, you are responsible for planning the annual sales meeting. You expect 300 people to attend the three day conference. For part of this time, the sales force will meet in small groups. You believe that groups of six would be a good size. How many groups would this be?

Input	Display
[3] [Ø] [Ø] [/] [6] [ENTER]	300/6 50.

On second thought you decide that groups containing an odd number of participants might be more effective. Recall your last entry using the [◄] arrow.

Input	Display
[◄]	300/6_

To calculate the new number of groups you must replace the six with an odd number. Five seems to make more sense than seven. Because you recalled using the [◄] arrow, the cursor is positioned at the end of the display. Use the [◄] to move the cursor one space to the left.

Input	Display
[◄]	300/6

Notice that after you move the cursor it becomes a flashing block ▮. Whenever you position the cursor "on top of" an existing character, it will be displayed as the flashing cursor.

Type in a 5 to replace the 6. One caution in replacing characters—once you type a new character over an existing character, the original is gone forever! You cannot recall an expression that has been typed over.

Input	Display
[5]	300/5_
[ENTER]	60.

Sixty seems like a reasonable number of groups, so you decide that each small group will consist of five participants.

Recall is also useful to verify your last entry, especially when your results do not seem to make sense. For instance, suppose you had performed this calculation:

Input	Display
[CLS] [3] [Ø] [/] [5] [ENTER]	30/5 6.

Even a tired, overworked manager like you realizes that 6 does not seem to be a reasonable result when you are dealing with hundreds of people! Recall your entry using the [►].

Input	Display
[►]	30/5

Because you recalled using the [►] the flashing cursor is now positioned over the first character in the display. To correct this entry you wish to insert an added zero. Using the [►], move the cursor until it is positioned over the zero. When making an INSert, you position the flashing cursor over the character before which you wish to make the insertion.

Input	Display
[►]	30/5

Use the INSert key to make space for the needed character.

Input	Display
[INS]	3 0/5

Pressing INSert moves all the characters one space to the right, and inserts a bracketed open slot. The flashing cursor is now positioned over this open space, indicating the location of the next typed input. Type in your zero. Once the entry is corrected, display your new result.

Input	Display
[0]	300/5
[ENTER]	60.

On the other hand, suppose that you had entered this calculation:

Input	Display
CLS 3 Ø Ø Ø / 5 ENTER	3000/5 600.

The results seem much too large. If you only have 300 people attending the meeting, how could you have 600 "small groups"? Recall your entry using the ► .

Input	Display
►	3000/5

The flashing cursor is now positioned over the first character in the display. To correct this entry eliminate one of the zeros. Using the ► move the cursor to the first zero (or any zero). When deleting a character, you position the cursor "on top of" the character to be deleted.

Input	Display
►	3000/5

Now use the DELete key to get rid of one of the zeros.

Input	Display
DEL	300/5

Pressing DELete causes all the characters to shift one space to the left. It deletes the character it is "on top of" and the space the character occupies. The flashing cursor stays in the same position indicating the next location for input. Since you have no other changes to make, complete the calculation.

Input	Display
ENTER	300/5 60.

(Note: Pressing the SPaCe key, when the cursor is positioned over a character, replaces the character leaving a blank space. DELete eliminates the character and the space it occupied.)

Errors

Recalling your last entry is essential when you get the dreaded ERROR message. Let us imagine that, unintentionally, you typed this entry into the PC-1350:

Input	Display
CLS 3 Ø Ø / / 5 ENTER	300//5 ERROR 1

Naturally you are surprised when this message appears! ERROR 1 is simply the computer's way of saying, "I don't know what you want me to do here".
At this time, when the ◄ key is pressed, the flashing cursor appears at the location where the error occurred.

Input	Display
◄	300//5

To correct this error use the DELete key.

Input	Display
DEL ENTER	300/5 60.

If, upon recalling your entry after an ERROR 1, you find that you have omitted a character, use the INSert sequence to correct it.
When using the PC-1350 as a calculator, the majority of the errors you encounter will be ERROR 1 (an error in syntax). For a complete listing of error messages, see APPENDIX A.

Serial Calculations

The PC-1350 allows you to use the results of one calculation as part of the following calculation.
Part of your responsibility in planning this conference is to draw up a detailed budget for approval. You know that your total budget is $150.00 for each attendant. Figure your total budget:

Input	Display
CLS 3 Ø Ø * 1 5 Ø ENTER	300*150 45000.

Of this amount you plan to use 15% for the final night's awards presentation. When performing serial calculations it is not necessary to retype your previous results, but DO NOT CLear between entries. What is the awards budget?

Input	Display
[*] [.] [1] [5]	45000.*.15_

Notice that as you type in the second calculation (* . 15), the computer automatically displays the result of your first calculation at the left of the screen and includes it in the new calculation. In serial calculations the entry must begin with an operator. As always, you end the entry with [ENTER] :

NOTE: The [%] key can not be used in the calculation. The [%] key should be used as a character only.
Example: 45000 [*] 15 [SHIFT] [%] →ERROR 1

Input	Display
[ENTER]	6750.

Continue allocating your budget. The hotel will cater your dinner for $4000:

Input	Display
[−] [4] [0] [0] [0]	6750.−4000_
[ENTER]	2750.

Decorations will be $1225:

Input	Display
[−] [1] [2] [2] [5] [ENTER]	1525.

Finally, you must allocate $2200 for the speaker and entertainment:

Input	Display
[−] [2] [2] [0] [0] [ENTER]	−675.

Obviously, you will have to change either your plans or your allocation of resources!

Negative Numbers

Since you want the awards dinner to be really special, you decide to stay with the planned agenda and spend the additional money. However, you wonder what percentage of the total budget will be used up by this item. First, change the sign of the remaining sum:

Input	Display
[*] [−] [1]	−675.*−1_
[ENTER]	675.

Now you add this result to your original presentation budget:

Input	Display
[+] [6] [7] [5] [0] [ENTER]	7425.

Dividing by 45000 gives you the percentage of the total budget this new figure represents:

Input	Display
[/] [4] [5] [0] [0] [0] [ENTER]	0.165

Fine, you decide to allocate 16.5% to the awards presentation.

Compound Calculations and Parentheses

In performing the above calculations, you could have combined several of these operations into one step. For instance, you might have typed both these operations on one line:

675+6750/45000

Compound calculations, however, must be entered very carefully:

675+6750/45000 might be interpreted as

$$\frac{675+6750}{45000} \quad \text{or} \quad 675+\frac{6750}{45000}$$

When performing compound calculations, the PC-1350 has specific rules of expression evaluation and operator priority (see APPENDIX D). Be sure you get the calculation you want by using parentheses to clarify your expressions:

(675+6750)/45000 or 675+(6750/45000)

To illustrate the difference that the placement of parentheses can make, try these two examples:

Input	Display
(6 7 5 + 6 7 5 Ø) / 4 5 Ø Ø Ø ENTER	0.165
6 7 5 + (6 7 5 Ø / 4 5 Ø Ø Ø) ENTER	675.15

Using Variables in Calculations

The PC-1350 can store up to 26 simple numeric variables under the alphabetic characters A to Z. If you are unfamiliar with the concept of variables, they are more fully explained in Chapter 4. You designate variables with an Assignment Statement:

```
A=5
B=-2
```

You can also assign the value of one variable (right) to another variable (left):

```
C=A+3
D=E
```

A variable may be used in place of a number in any calculation.

Now that you have planned your awards dinner, you need to complete arrangements for your conference. You wish to allocate the rest of your budget by percentages also. First you must find out how much money is still available. Assign a variable (R) to be the amount left from the total:

Input	Display
R = 4 5 Ø Ø Ø − 7 4 2 5 ENTER	R=45000−7425_ 37575.

As you press ENTER the PC-1350 performs the calculation and displays the new value of R. You can display the current value of any variable by entering the alphabetic character it is stored under:

Input	Display
R ENTER	37575.

You can then perform calculations using your variable. The value of (R) will not change until you assign it a new value.

You wish to allocate 60% of the remaining money to room rental:

Input	Display
R * . 6 Ø ENTER	R*.60_ 22545.

Similarly, you want to allocate 25% of your remaining budget to conduct management training seminars:

Input	Display
R * . 2 5 ENTER	9393.75

Variables will retain their assigned values even if the machine is turned OFF or undergoes an AUTO OFF. Variables are lost only when:

* You assign a new value to the same variable.
* You type in CLEAR ENTER (not the Clear key).
* You clear the machine using the ALL RESET button.
* The batteries are changed.

There are certain limitations on the assignment of variables, and certain programming procedures which cause them to be changed. See Chapter 4 for a discussion of assignment. See Chapter 5 for a discussion of the use of variables in programming.

Chained Calculations

In addition to combining several operators in one calculation, the PC-1350 also allows you to perform several calculations one after the other—without having to press ENTER before moving on. You must separate the equations with commas. Only the result of the final calculation is displayed. (Remember too, that the maximum line length accepted by the computer is 80 characters including ENTER .)

You wonder how much money would have been available for rooms if you had kept to your original allocation of 15% for the awards dinner:

Input	Display
R = . 8 5 * 4 5 Ø Ø Ø , R * . 6 Ø	R=.85*45000, R*.60_

Although the computer performs all the calculations in the chain, it displays only the final result:

Input	Display
ENTER	R=.85*45000, R*.60 22950.

To find the value of R used in this calculation, enter R:

Input	Display
R ENTER	38250.

Scientific Notation

People who need to deal with very large and very small numbers often use a special format called exponential or scientific notation. In scientific notation a number is broken down into two parts.

The first part consists of a regular decimal number between 1 and 10. The second part represents how large or small the number is in powers of 10.

As you know, the first number to the left of the decimal point in a regular decimal number shows the number of 1's, the second shows the number of 10's, the third the number of 100's, and the fourth the number of 1000's. These are simply increasing powers of 10:

$10^0=1$, $10^1=10$, $10^2=100$, $10^3=1000$, etc.

Scientific notation breaks down a decimal number into two parts: one shows what the numbers are, the second shows how far a number is to the left, or right, of the decimal point. For example:

1234 becomes 1.234 times 10^3 (3 places to the right)
654321 becomes 6.54321 times 10^5 (5 places to the right)
.000125 becomes 1.25 times 10^{-4} (4 places to the left)

Scientific notation is useful for many shortcuts. You can see that it would take a lot of writing to show 1.0 times 10^{87}—a 1 and 87 zeros! But, in scientific notation this number looks like this:

1.0×10^{87} or 1.0E 87

The PC-1350 uses scientific notation whenever numbers become too large to display using decimal notation. This computer uses the capital letter E to mean "times ten to the":

1234567890000 is displayed as 1.23456789 E 12
.000000000001 is displayed as 1. E −12

Those of you who are unfamiliar with this type of notation should take some time to put in a few very large and very small numbers to note how they are displayed.

Limits

The largest number which the PC-1350 can handle is ten significant digits, with two digit exponents. In other words the largest number is:

9.999999999 E 99= 9999999990000000000

Press [CLS], then the [↓] or [↑] key. If you operated these keys just after completing the calculation example on the previous page, you should see "35." in your display. The numeric data displayed is the result of your last calculation.

The PC-1350 can "remember" the last answer (result) obtained through manual calculation, and recall it on its display with the [↓] or [↑] key.

In the case of the serial calculation described above, you could use the result of the previous calculation only as the first member of the subsequent calculation formula. With the last answer feature, however, you can place the result of the previous calculation in any position of the subsequent calculation.

(Example) Use the result (6.25) of the operation, 50÷8, to compute 12×5÷6.25+24×3÷6.25 =:

Input	Display
50 [/] 8 [ENTER]	50/8 6.25 (Last answer)
12 [*] 5 [/] [↑] (or [↓])	12*5/6.25_ (Last answer recalled)
[+] 24 [*] 3 [/] [↑] (or [↓])	12*5/6.25+24*3/6.25_ (Last answer recalled)
[ENTER]	21.12
[CLS] [↓]	21.12_

The last answer is replaced with the result of the previous calculation by performing a manual calculation with the [ENTER] key.

As shown in this example, the last answer can be recalled anytime and anyplace, but will be replaced with a new last answer resulting from the last calculation.

The last answer is not cleared by the [CLS] or key operation.

- The last answer cannot be recalled when the computer is not in the RUN mode, program execution is temporarily halted, or the Trace mode is selected.

Length of Formula

The length of a formula you can put into your computer has a certain limitation. With the PC-1350, up to 79 key strokes can be used to enter a single calculation formula (excluding the [ENTER] key). If you attempt the 80th key stroke, the cursor (▮) will start blinking on that character, indicating that the 80th key entry is not valid.

Scientific Calculations

The PC-1350 is equipped with the basic functions shown below. Note that the notation of the functions in BASIC may differ from conventional mathematical notations.

Function	Conventional notation	Key operation	Remarks
Trigonometric functions	sin cos tan	SIN COS TAN	
Inverse trigonometric functions	$\sin^{-1}$ $\cos^{-1}$ $\tan^{-1}$	ASN ACS ATN	
Common logarithm	log	LOG	$\log_{10}x$ (logarithm based on 10)
Natural logarithm	ln	LN	$\log_e x$ (logarithm based on e)
Exponential function	e^x	EXP	e≃2.718281828
Exponentiation		^	A^B for A ^ B
Square root	$\sqrt{}$	$\sqrt{}$ or SQR	$\sqrt[3]{8}$ → 8^(1/3)
Degrees (decimal) → degrees (degrees, minutes, seconds) conversion		DMS	Angle conversion (Do not leave out the 0 as in DEG.5 instead of DEG 0.5)
Degrees (degrees, minutes, seconds) → degrees (decimal) conversion		DEG	
Integer		INT	In INT(x), obtains the largest integer less than or equal to x.
Absolute	|X|	ABS	In ABS(x), obtains the absolute value of x.
Signum		SGN	Results in 1 when x>0, −1 when x<0, and 0 when x=0 for SGN(x).
Pi	π	π or PI	π≃3.141592654
Hexadecimal → decimal notation		&	Converts x to a number in base 10 for &x.

Angular unit	Command	Description
Degree	DEGREE	Represents a right angle as 90[°].
Radian	RADIAN	Represents a right angle as π/2[rad].
Grade	GRAD	Represents a right angle as 100[g].

These instructions are used to specify angular units in program. For practice, use these instructions to specify angular units in the following calculation examples:

(Example) sin 30°=

(Operation) DEGREE [ENTER] (Specifies "degree" for angular unit.)

SIN 30 [ENTER] | **0.5**

(Example) $\tan\frac{\pi}{4}=$

(Operation) RADIAN [ENTER] (Specifies "radian" for angular unit.)

TAN (PI/4) [ENTER] | **1.**

(Example) $\cos^{-1}(-0.5)=$

(Operation) DEGREE [ENTER] (Specifies "degree" for angular unit.)

ACS−0.5 [ENTER] | **120.**

(Example) log 5+ln 5=

(Operation) LOG 5+LN 5 [ENTER] | **2.308407917**

(Example) $e^{2+3}=$

(Operation) EXP (2+3) [ENTER] | **148.4131591**

(Example) $\sqrt{4^3 + 6^4} =$

(Operation) √(4^3 + 6^4) [ENTER] | **36.87817783**

(Example) Convert 30 deg. 30 mm. in sexagenary notation into decimal notation.

(Operation) DEG 30.30 [ENTER] | **30.5**
(30.5 degree)

(Example) Convert 30.755 deg. in decimal notation into sexagenary notation

(Operation) DMS 30.755 [ENTER] | **30.4518**
(30 deg. 45 min. 18 sec.)

(Example) Convert CF8 to its decimal equivalent.

(Operation) &CF8 [ENTER] | **3320.**

Priority in Manual Calculation

You can type in formulas in the exact order in which they are written, including parentheses or functions. The order of priority in calculation and treatment of intermediate results will be taken care of by the computer itself.

The internal order of priority in manual calculation is as follows:

1) Recalling variables or π.
2) Function (sin, cos, etc.)
3) Power (^)
4) Sign (+, −)
5) Multiplication or division (*, /)
6) Addition or subtraction (+, −)
7) Comparison of magnitude (>, >=, <, <=, <>,=)
8) Logical AND, OR

Notes: * If parentheses are used in a formula, the operation given within the parentheses has the highest priority.
* Composite functions are operated from right to left (sin $\cos^{-1}$ Ø.6).
* Chained power (3^{4^2} or 3^4^2) is operated from right to left.
* For the above items 3) and 4), the last entry has a higher priority.
(e.g.) −2^4 → $-(2^4)$
3^−2 → 3^{-2}

Printing for Manual Calculations

The calculation steps and results can be printed when the optional printer (CE-126P) is connected and turned on, and [SHIFT] [P↔NP] are pressed. (Print Mode)
If a printout is not desired, either turn off the CE-126P or press [SHIFT] and [P↔NP] again. (Non-Print Mode)

Caution

Calculation Error

The following types of errors occur in ordinary calculators, pocket computers, and personal computers.

(1) Errors due to least significant digit processing

Usually, the maximum number of digits that can be calculated in a computer is fixed. For example, 4÷3 results in 1.3333333333... In a computer with a maximum of 8 digits, the 8 digits are significant digits; other least significant digits are either truncated or rounded.

(example) computer with 1Ø significant digits

4/3 [ENTER] → 1.3333333333... (10 significant digits; Truncated, rounded)

Therefore, the calculated result differs from the true value by the amount truncated or rounded. (This difference is the error.)
In the unit, a 12-digit calculated result is obtained. This result is rounded and specially processed to minimize error in the displayed value.

(example) 4÷3×3

① 4/3*3 [ENTER] → 4. Calculated in succession

② 4/3 [ENTER] → 1.333333333
*3 [ENTER] → 3.999999999
Calculated in dependently

* When calculated in succession, the result of 4÷3 is obtained internally in 12 digits and is used for calculation and then rounded.
When calculated independently, the displayed value (1Ø digits) is used for the calculation.

(2) Error due to an approximation algorithm

Since the function employs an approximation algorithm, the error in the computer is slightly larger than those for regular (non-function) calculations.
Although the unit performs a variety of processing to minimize error in the displayed value, the errors add up especially around singular points and inflection points of functions.
And also, when performing a succession of function calculations such as power calculations, the errors add up and may appear on the display.

(example) $6Ø^6$ =

6Ø^6 [ENTER] → 4.665599999E1Ø

Although $6Ø^6$ is $4.6656 \times 1Ø^{10}$, the unit calculates it (y^x) based on the following power expression.

$y^x = 1Ø^{x \times \log y}$

In other words, $6Ø^6$ is calculated as $1Ø^{6 \times \log 60}$.

CHAPTER 4
CONCEPTS AND TERMS OF BASIC

In this Chapter we will examine some concepts and terms of the BASIC language.

String Constants

In addition to numbers, there are many ways that the SHARP PC-1350 uses letters and special symbols. These letters, numbers, and special symbols are called characters.
These characters are available on the PC-1350:

1 2 3 4 5 6 7 8 9 0
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z
! " # $ % & ( ) * + , − . / : ; <=> ? @ √ π ^

In BASIC, a collection of characters is called a string. In order for the PC-1350 to tell the difference between a string and other parts of a program, such as verbs or variable names, you must enclose the characters of the string in quotation marks (").
The following are examples of string constants:

"HELLO"
"Goodbye"
"SHARP PC-1350"

The following are not valid string constants:

"COMPUTER No ending quote
"ISN"T" Quote can't be used within a string

Hexadecimal Numbers

The decimal system is only one of many different systems to represent numbers. Another which has become quite important when using computers is the hexadecimal system. The hexadecimal system is based on 16 instead of 10. To write hexadecimal numbers you use the familiar 0 ~ 9 and 6 more "digits": A,B,C,D,E, and F. These correspond to 10, 11, 12, 13, 14, and 15. When you want the PC-1350 to treat a number as hexadecimal put an ampersand '&' character in front of the numeral:

&A = 10
&10 = 16
&100 = 256
&FFFF = 65535

Variables

Computers are made up of many tiny memory areas called bytes. Each byte can be

thought of as a single character. For instance, the word byte requires four bytes of memory because there are four characters in it. To see how many bytes are available for use, simply type in MEM [ENTER] . The number displayed is the number of bytes available for writing programs. This technique works fine for words, but is very inefficient when you try to store numbers. For this reason, numbers are stored in a coded fashion. Thanks to this coding technique, your computer can store large numbers in only eight bytes. The largest number that can be stored is +9.999999999E+99.
The smallest number is +1.E−99. This gives you quite a range to choose from. However, if the result of a calculation exceeds this range, the computer will let you know by displaying an error message on the screen. For the error message refer to Appendix A. To see it right now type in:

9 E 99*9 [ENTER]

To get the computer working properly again, just press the [CLS] key. But how do you go about storing all this information? It's really very easy. The computer likes to use names for different pieces of data. Let's store the number 556 into the computer. You may call this number by any name that you wish, but for this exercise, let's use the letter R. The statement, LET, can be used to instruct the computer to assign a value to a variable name but only in a program statement. However, the LET command is not necessary, so we will not use it very often. Now, type in R=556 and press the [ENTER] . The computer now has the value 556 associated with the letter R. These letters that are used to store information are called variables. To see the content of the variable R, press the [CLS] key, the R key and the [ENTER] key. The computer responds by showing you the value 556 on the right of your screen. This ability can become very useful when you are writing programs and formulas.

Next, let's use the R variable in a simple formula. In this formula, the variable R stands for the radius of a circle whose area we want to find. The formula for the area of a circle is: $A=\pi *R^2$. Type in R [SHIFT] [^] 2 [*] [SHIFT] [π] [ENTER] The result is 971179.3866. This technique of using variables in equations will become more understandable as we get into writing programs.

So far, we've only discussed numeric variables. What about storing alphabetic characters? Well, the idea is the same, but, so the computer will know the difference between the two kinds of variables, add a $ to the variable name. For instance, let's store the word BYTE in the variable B$. Notice the $ after the B!

This tells the computer that the contents of the letter B is alphabetic, or string data.

To illustrate this, key in B [SHIFT] [$] = [SHIFT] ["] BYTE [SHIFT] ["] [ENTER] . The value BYTE is now stored in the variable B$. To make sure of this, press the [CLS] key and type in B [SHIFT] [$] [ENTER] . The screen shows BYTE. This time the display is on the left side of the screen, instead of the right.

Note: The contents of character strings or character variables are displayed from the left edge of the first line.

Variables handled by the SHARP PC-1350 are divided into the followings:

- Variables
  - Numeric variables
    - Fixed numeric variables (A to Z)
    - Simple numeric variables (AB, C1, etc.)
    - Numeric array variables
  - String variables
    - Fixed character variables (A$ to Z$)
    - Simple character variables (BB$, C2$, etc.)
    - Character array variables

Fixed Variables

The first section, fixed variable, is always used by the computer for storing data. It can be thought of as pre-allocated variable space. In other words, no matter how much memory your program uses up, you will always have at least 26 variables to choose from to store data in. This data can be one of two types: NUMERIC or STRING (alphabetic character). Fixed memory locations are eight bytes long and can be used for only one type of data at a time. To illustrate this, type in the following example:

A=123 [ENTER]
A$ [ENTER]

You get the message:

ERROR 9

This means that you have put numeric data into the area of memory called A and then told the computer to show you that information again as STRING data. This confuses the computer so it says that there is an error condition. Press the [CLS] key to clear error condition. Now try the following example:

A$="ABC" [ENTER]
A [ENTER]

Again, the computer is confused and gives the ERROR 9 message. Look at Figure shown below to see that the variable name A equals the same area in memory as the variable name A$, and that B equals B$, and so on for all the letters of the alphabet.
Figure:

A = A$ = A(1)	=	A$(1)
B = B$ = A(2)	=	A$(2)
C = C$ = A(3)	=	A$(3)
D = D$ = A(4)	=	A$(4)
E = E$ = A(5)	=	A$(5)
F = F$ = A(6)	=	A$(6)
G = G$ = A(7)	=	A$(7)

H = H$ = A(8)	=	A$(8)
I = I$ = A(9)	=	A$(9)
J = J$ = A(10)	=	A$(10)
K = K$ = A(11)	=	A$(11)
L = L$ = A(12)	=	A$(12)
M = M$ = A(13)	=	A$(13)
N = N$ = A(14)	=	A$(14)
O = O$ = A(15)	=	A$(15)
P = P$ = A(16)	=	A$(16)
Q = Q$ = A(17)	=	A$(17)
R = R$ = A(18)	=	A$(18)
S = S$ = A(19)	=	A$(19)
T = T$ = A(20)	=	A$(20)
U = U$ = A(21)	=	A$(21)
V = V$ = A(22)	=	A$(22)
W = W$ = A(23)	=	A$(23)
X = X$ = A(24)	=	A$(24)
Y = Y$ = A(25)	=	A$(25)
Z = Z$ = A(26)	=	A$(26)

Simple Variables

Simple variable names are specified by two (or more) alphanumeric characters, such as AA or B1. Unlike fixed variables, simple variables have no dedicated storage area in the memory. The area for simple variables is automatically set aside (within the program and data area) when a simple variable is first used.
Since separate memory areas are defined for simple numeric variables and simple character variables even if they have the same name, variables such as AB and AB$, for example, may be used at the same time.
While alphanumeric characters are usable for simple variable names (as for alphabetic characters, only upper case characters are usable), the first character of a variable name must always be an alphabetic character. If more than two characters are used to define a variable name, only the first two characters are meaningful.

Note:
- The function or BASIC instruction names for the PC-1350 computer are not usable as variable names. (e.g.) PI, IF, TO, ON, SIN, etc.
- Each simple character variable can hold up to 16 characters or symbols.

Array Variables

For some purposes it is useful to deal with numbers as an organized group, such as a

list of scores or a tax table. In BASIC these groups are called arrays. An array can be either one-dimensional, like a list, or two-dimensional, like a table.

To define an array, the DIM (short for dimension) statement is used. Arrays must always be "declared" (defined) before they are used. (Not like the single-value variables we have been using.) The form for the numeric DIMension statement is:

DIM numeric-variable-name (size)

where:

numeric-variable-name is a variable name which conforms to the normal rules for numeric variable names previously discussed.

size is the number of storage locations and must be a number in the range 0 through 255. Note that when you specify a number for the size, you get one more location than you specified.

Examples of legal numeric DIMension statements are:

```
DIM X (5)
DIM AA (24)
DIM Q5 (0)
```

The first statement creates an array X with 6 storage locations. The second statement creates an array AA with 25 locations. The third statement creates an array with one location and is actually rather silly since (for numbers at least), it is the same as declaring a single-value numeric variable.

It is important to know that an array-variable X and a variable X are separate and distinct to the PC-1350. The first X denotes a series of numeric storage locations, and the second a single and different location.

Now that you know how to create arrays, you might be wondering how it is that we refer to each storage location. Since the entire group has only one name, the way in which we refer to a single location (called an "element") is to follow the group name with a number in parentheses. This number is called a "subscript". Thus, for example, to store the number 8 into the fifth element of our array X (declared previously) we would write:

```
X(4)=8
```

If the use of 4 is puzzling, remember that the numbering of elements begins at zero and continues through to the number of elements declared in the DIM statement.

The real power of arrays lies in the ability to use an expression or a variable name as a subscript.

To declare a character array a slightly different form of the DIM statement is used:

DIM character-variable-name (size) * length

where:

character-variable-name is a variable name which conforms to the rules for normal character variables as discussed previously.

size is the number of storage locations and must be in the range 0 through 255. Note that when you specify a number, you get one more location than you specified.

*length is optional. If used, it specifies the length of the strings that comprise the array. Length is a number in the range 1 to 80. If this clause is not used, the strings will have the default length of 16 characters.

Example of legal character array declarations are:

```
DIM X$(4)
DIM NM$(10)*10
DIM IN$(1)*80
DIM R$(0)*26
```

The first example creates an array of five strings each able to store 16 characters. The second DIM statement declares an array NM with eleven strings of 10 characters each.

Explicit definition of strings smaller than the default helps to conserve memory space. The third example declares a two element array of 80-character strings and the last example declares a single string of twenty-six characters.

Besides the simple arrays we have just studied, the PC-1350 allows "two-dimensional" arrays. By analogy, a one-dimensional array is a list of data arranged in a single column. A two-dimensional array is a table of data with rows and columns.

The two-dimensional array is declared by the statement:

DIM numeric-variable-name (rows, columns)

or

DIM character-variable-name (rows, columns)*length

where:

rows specifies the number of rows in the array. This must be a number in the range 0 through 255. Note that when you specify the number of rows you get one more row than the specification.

columns specifies the number of columns in the array. This must be a number in the range 0 through 255. Note that when you specify the number of columns you get one more column than the specification.

The following diagram illustrates the storage locations that result from the declaration DIM T (2, 3) and the subscripts (now composed of two numbers) which pertain to each storage location:

	column 0	column 1	column 2	column 3
row 0	T (0,0)	T (0,1)	T (0,2)	T (0,3)
row 1	T (1,0)	T (1,1)	T (1,2)	T (1,3)
row 2	T (2,0)	T (2,1)	T (2,2)	T (2,3)

Note: Two-dimensional arrays can rapidly eat up storage space. For example, an array with 25 rows and 35 columns uses 875 storage locations!

Arrays are very powerful programming tools.

The following table shows the number of bytes used to define each variable and the number used by each program statement.

Variable	Variable name	Data	
Numeric variable	7 bytes	8 bytes	
String variable	7 bytes	Array variable	Specified number*
		Simple variable (two-character variable)	16 bytes

* For example, if DIM Z$(2, 3)* 10 is specified, 12 variables, each capable of storing 10 characters, are reserved. This requires 7 bytes (variable name) + 10 bytes (number of characters)×12=127 bytes.

Element	Line number	Statement & function	ENTER , and others
Number of bytes used	3 bytes	1 byte	1 byte

Variables in the Form of A( )

While a data area on the computer's memory is set aside for fixed variables, it may also be used to define subscripted variables which have the same form as array variables.

There are 26 fixed variable names available: i.e. A through Z(A$ through Z$). Each of these names can be subscripted with the numbers 1 through 26, such as A(1)–A(26) or A$(1)–A$(26). This means that variable A(1) may be used in place of variable A, A(2) in place of B, A(3) in place of C, and so forth.

However, if an array named A or A$ has already been defined by the DIM statement, subscripted variables named A cannot be defined. For example, if an array A is defined by DIM A(5) the location for A(0) through A(5) are set aside in the program/data area. So if you specify variable A(2), it does not refer to the fixed variable B, but refers to the array variable A(2) defined in the program/data area. If you specify A(9), it will cause an error since A(9) is outside the range of the dimension specified by the DIM A(5) statement.

On the other hand, if subscripted variables are already defined in the form of A( ), it is not possible to define arrays A or A$ by using the DIM statement, unless the definition for the subscripted variables is cleared with the CLEAR statement.

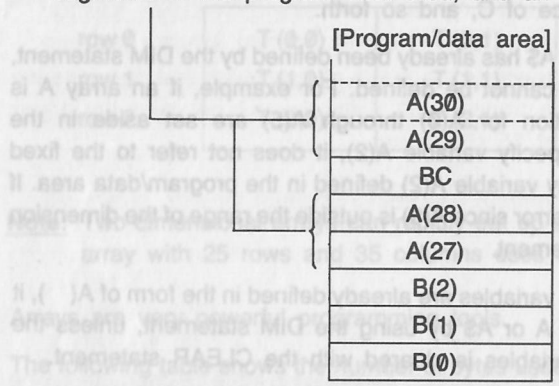
* Using subscripts in excess of 26:
  If subscripts greater than 26 are used for subscripted variables A( ) when array A is not defined by a DIM statement, the corresponding locations in the program/data area are set aside for these A( ) variables. For instance, if you execute A(35)=5, locations for variables A(27) to A(35) will be reserved in the program/data area.

While variables subscripted in excess of 26 are treated as array variables, they are subject to the following special restrictions:

(1) Locations for an array with the same name must be contiguous in the program/data area. Otherwise, an error will occur.

```
10 DIM B(2)
20 A(28)=5
30 BC=12
40 A(30)=9
```

If this program is excuted, the array named "A" is not defined in two consecutive segments in the program data area, and an error will result at line 40.



(2) Numeric array variables and character array variables with the same subscript cannot be defined at the same time. For example, A(30) and A$(30) cannot be defined at the same time, since they use the same location in the program/data area.

(3) Two dimensional arrays cannot be defined, nor is it possible to specify the length of character strings to be held in character array variables. For example, the length of a character string which can be held in the character array variable A$( ) is limited to seven characters or less.

(4) Variables subscripted with zero (0) cannot be defined. If A(0) or A$(0) is defined, an error will result.

(5) When A(27), or A$(27) and higher is first used, 7 bytes are used for the variable names and 8 bytes are used for each variable.

Expressions

An expression is some combination of variables, constants, and operators which can be evaluated to a single value. The calculations which you entered in Chapter 3 were examples of expressions. Expressions are an intrinsic part of BASIC programs. For example, an expression might be a formula that computes an answer to some equation, a test to determine the relationship between two quantitites, or a means to format a set of strings.

Numeric Operators

The PC-1350 has five numeric operators. These are the arithmetic operators which you used when exploring the use of the PC-1350 as a calculator in Chapter 3: Chapter 3:

+ Addition
− Subtraction
* Multiplication
/ Division
^ Power

A numeric expression is constructed in the same way that you entered compound calculator operations. Numeric expressions can contain any meaningful combination of numeric constants, numeric variables, and the numeric operators:

```
(A*B)^2
A(2,3)+A(3,4)+5.0−C
(A/B)*(C+D)
```

String Expressions

String expressions are similar to numeric expressions except that there is only one string operator—concatenation (+). This is the same symbol used for plus. When used with a pair of strings, the + attaches the second string to the end of the first string and makes one longer string. You should take care in making more complex string concatenations and other string operations because the work space used by the PC-1350 for string calculations is limited to only 80 characters.

Note: String quantities and numeric quantities cannot be combined in the same expression unless one uses one of the functions which convert a string value into a numeric value or vice versa:

"15" + 10	is illegal
"15" + "10"	is "1510", not "25"

Relational Expressions

A relational expression compares two expressions and determines whether the stated relationship is true or false. The relational operators are:

>	Greater Than
> =	Greater Than or Equal To
=	Equals
<>	Not Equal To
< =	Less Than or Equal To
<	Less Than

The following are valid relational expressions:

A<B
C(1,2)>=5
D(3)<>8

If A was equal to 10, B equal to 12, C(1, 2) equal to 6, and D(3) equal to 9, all of these relational expressions would be true.

Character strings can also be compared in relational expressions. The two strings are compared character by character according to their ASCII value starting at the first character (see Appendix B for ASCII values). If one string is shorter than the other, a 0 or NUL will be used for any missing positions. All of the following relational expressions are true:

"ABCDEF"="ABCDEF"
"ABCDEF" <> "ABCDE"
"ABCDEF" > "ABCDE"

Relational expressions evaluate to either true or false. The PC-1350 represents true by a 1; false is represented by a 0. In any logical test an expression which evaluates to 1 or more will be regarded as true while one which evaluates to 0 or less will be considered false. Good programming practice, however, dictates the use of an explicit relational expression instead of relying on this coincidence.

Logical Expressions

Logical expressions are relational expressions which use the operators AND, OR, and NOT. AND and OR are used to connect two relational expressions; the value of the combined expression is shown in the following tables:

A AND B

Value of B \ Value of A	True	False
True	True	False
False	False	False

A OR B

Value of B \ Value of A	True	False
True	True	True
False	True	False

(Note: Value of A and B must be 0 or 1)

- Decimal numbers can be expressed in the binary notation of 16 bits as follows:

DECIMAL NOTATION	BINARY NOTATION OF 16-BIT
32767	0111111111111111
3	0000000000000011
2	0000000000000010
1	0000000000000001
0	0000000000000000
−1	1111111111111111
−2	1111111111111110
−3	1111111111111101
−32768	1000000000000000

The negative (NOT) of a binary number 0000000000000001 is taken as follows:

NOT 0000000000000001
(Negative) → 1111111111111110

Thus, 1 is inverted to 0, and 0 to 1 for each bit, which is called "to take negative (NOT)".
Then, the following will result when 1 and NOT 1 are added together:

```
  0000000000000001 (1)
+) 1111111111111110 (NOT 1)
-------------------
  1111111111111111 (−1)
```

Thus, all bits become 1. According to the above number list, the bits become −1 in decimal notation, that is 1 + NOT 1=−1.
The relationship between numerical value X and its negative (NOT X) is:

X + NOT X=−1

This results in an equation of NOT X=−X−1
i.e. NOT X=−(X + 1)
From the equation the following are found to result.

NOT 0=−1
NOT −1=0
NOT −2=1

More than two relational expressions can be combined with these operators. You should take care to use parentheses to make the intended comparison clear.

(A<9) AND (B>5)
(C=5) OR (C=6) OR (C=7)

The PC-1350 implements logical operators as "bitwise" logical functions on 16 bit quantities. (See note on relational expressions and true and false). In normal operations this is not significant because the simple 1 and 0 (true and false) which result from a relational expression uses only a single bit. If you apply a logical operator to a value other than 0 or 1, it works on each bit independently. For example if A is 17, and B is 22, (A OR B) is 23:

17 in binary notation is 10001
22 in binary notation is 10110
17 OR 22 is 10111 (1 if 1 in either number, otherwise 0)
10111 is 23 in decimal.

If you are a proficient programmer, there are certain applications where this type of operation can be very useful. Beginning programmers should stick to clear, simple true or false relational expressions.

Parentheses and Operator Precedence

When evaluating complex expressions the PC-1350 follows a predefined set of priorities which determine the sequence in which operators are evaluated. This can be quite significant:

5+2*3 could be

5+2=7 or 2*3=6
7*3=21 6+5=11

The exact rules of "operator precedence" are given in Appendix D.

To avoid having to remember all these rules and to make your program clearer, always use parentheses to determine the sequence of evaluation. The above example is clarified by writing either:

(5+2)*3 or 5+(2*3)

RUN Mode

In general, any of the expressions described before can be used in the RUN mode as well as in programming a BASIC statement. In the RUN mode an expression is computed and displayed immediately. For example:

Input	Display
(5>3) AND (2<6) [ENTER]	1.

The 1 means that the expression is true.

Functions

Functions are special components of the BASIC language which take one value and transform it into another value. Functions act like variables whose value is determined by the value of other variables or expressions. ABS is a function which produces the absolute value of its argument:

ABS(−5) is 5
ABS(6) is 6

LOG is a function which computes the log to the base 10 of its argument.

LOG (100) is 2
LOG (1000) is 3

A function can be used any place that a variable can be used. Many functions do not require the use of parentheses:

LOG 100 is the same as LOG (100)

You must use parentheses for functions which have more than one argument. Using parentheses always makes programs clearer.

See Chapter 9 for a complete list of functions available on the PC-1350.

CHAPTER 5
PROGRAMMING THE PC-1350

In the previous chapter we examined some of the concepts and terms of the BASIC programming language. In this chapter you will use these elements to create programs on the PC-1350. Let us reiterate however, this is not a manual on how to program in BASIC. What this chapter will do is familiarize you with the use of BASIC on your PC-1350.

Programs

A program consists of a set of instructions to the computer. Remember the PC-1350 is only a machine. It will perform the exact operations that you specify. You, the programmer, are responsible for issuing the correct instructions.

BASIC Statements

The PC-1350 interprets instructions according to a predetermined format. This format is called a statement. You always enter BASIC statements in the same pattern. Statements must start with a line number:

```
10: INPUT A
20: PRINT A*A
30: END
```

Line Numbers

Each line of a program must have a unique line number—any integer between 1 and 65279. Line numbers are the reference for the computer. They tell the PC-1350 the order in which to perform the program. You need not enter lines in sequential order (although if you are a beginning programmer, it is probably less confusing for you to do so). The computer always begins execution with the lowest line number and moves sequentially through the lines of a program in ascending order.

When programming it is wise to allow increments in your line numbering (10, 20, 30, . . . 10, 30, 50 etc). This enables you to insert additional lines if necessary.

CAUTION: Do not use the same line numbers in different programs you plan to merge.

If you use the same line number, the oldest line with that number is deleted when you enter the new line.

BASIC Verbs

All BASIC statements must contain verbs. Verbs tell the computer what action to perform. A verb is always contained within a program, and as such is not acted upon immediately.

Some statements require or allow an operand:

```
10: DATA "HELLO"
20: READ B$
30: PRINT B$
40: END
```

Operands provide information to the computer telling it what data the verb will act upon. Some verbs require operands, with other verbs they are optional. Certain verbs do not allow operands. (See Chapter 9 for a complete listing of BASIC verbs and their use on the PC-1350.)

Note: Verbs, commands and functions must be typed in the upper case character mode.

BASIC Commands

Commands are instructions to the computer which are entered outside of a program. Commands instruct the computer to perform some action with your program or to set modes which affect how your programs are executed.

Unlike verbs, commands have immediate effects—as soon as you complete entering the command (by pressing the [ENTER] key), the command will be executed. Commands are not preceded by a line number:

```
RUN
NEW
RADIAN
```

Some verbs may also be used as commands. (See Chapter 9 for a complete listing of BASIC commands and their use on the PC-1350).

Modes

You will remember that when using the PC-1350 as a calculator, it is set in the RUN mode.

The RUN mode is also used to execute the programs you create.

The PROgram mode is used to enter and edit your programs.

The RESERVE mode enables you to designate and store predefined string variables and is used in more advanced programming (See Chapter 6).

Beginning to Program on the PC-1350

After all your practice in using the PC-1350 as a calculator you are probably quite at home with the keyboard. From now on, when we show an entry, we will not show every keystroke. Remember to use [SHIFT] to access characters above the keys and END EVERY LINE BY PRESSING THE [ENTER] KEY.

Now you are ready to program!
To enter program statements into the computer, the computer must first be placed in the PROGRAM mode using the [MODE] key. The display will appear as in the following illustration.

```
PRO   PROGRAM MODE
      >
```

Enter NEW command.

Input	Display
NEW	>

The NEW command clears the PC-1350 memory of all existing programs and data. The prompt appears after you press [ENTER], indicating that the computer is awaiting input.

Example 1—Entering and Running a Program

Make sure the PC-1350 is in the PRO mode and enter the following program:

Input	Display
10 PRINT "HELLO"	**10: PRINT "HELLO"**

Notice that when you push [ENTER] the PC-1350 displays your input, automatically inserting a colon (:) between the line number and the verb. Verify that the statement is in the correct format.

Now change the mode to the RUN:

Input	Display
RUN	**HELLO**

Since this is the only line of the program, the computer will stop executing at this point. Press [ENTER] to get out of the program and reenter RUN if you wish to execute the program again.

Example 2—Editing a Program

Suppose you wanted to change the message that your program was displaying, that is, you wanted to edit your program. With a single line program you could just retype the entry, but as you develop more complex programs editing becomes a very important component of your programming. Let's edit the program you have just written.

Are you still in the RUN mode? If so switch back to the PROgram mode.

You need to recall your program in order to edit it. Use the Up Arrow [↑] to recall your program. If your program was completely executed, the [↑] will recall the last line of the program. If there was an error in the program, or if you used the BREAK([BRK]) key to stop execution, the [↑] will recall the line in which the error or BREAK occurred. To make changes in your program use the [↑] to move up in your program (recall the previous line) and the [↓] to move down in your program (display the next line). If held down, the [↑] and the [↓] will scroll vertically, that is, they will display each line moving up or down in your program.

You will remember that to move the cursor within the program line displayed at the top line of the display you use the [►] (right arrow) and [◄] (left arrow). Using the [►] position the cursor over the first character you wish to change:

Note: Even if several lines of a program are displayed on the display unit, the cursor can be moved only within the first displayed line. To edit a lower line, move the line to the top using the [↓] key and then edit.

Input	Display
[↑]	10: PRINT "HELLO"
[◄] [◄] [◄] [◄] [◄]	10 PRINT "HELLO"

Notice that the cursor is now in the flashing block form indicating that it is "on top of" an existing character. Type in:

Input	Display
GOODBYE"!	10 PRINT "GOODBYE"!_

Don't forget to press [ENTER] at the end of the line. Change into the RUN mode.

Input	Display
RUN [ENTER]	ERROR 1 IN 10

This is a new kind of error message. Not only is the error type identified (our old friend the syntax error) but the line number in which the error occurs is also indicated.

Press the [CLS] to clear the error condition.
And return to the PRO mode. You must be in the PROgram mode to make changes in a program. Using the [↑] (or [↓]), recall the line in which the error occurred.

Input	Display
[↑] (or [↓])	10: PRINT "GOODBYE"!

The flashing cursor is positioned over the problem area. In Chapter 4 you learned that when entering string constants in BASIC all characters must be contained within quotation marks. Use the DELete key to eliminate the "!":

Input	Display
[DEL]	10 PRINT "GOODBYE"_

Now let's put the ! in the correct location. When editing programs, DELete and INSert are used in exactly the same way as they are in editing calculations (See Chapter 3). Using the [◄] position the cursor on top of the character which will be the first character following the insertion.

Input	Display
[◄]	10 PRINT "GOODBYE"

Press the INSert key. A □ will indicate the spot where the new data will be entered:

Input	Display
[INS]	10 PRINT "GOODBYE □"

Type in the !. The display looks like this:

Input	Display
!	10 PRINT "GOODBYE!"

Remember to press [ENTER] so the correction will be entered into the program.

Note 1. If you wish to DELete an entire line from your program just type in the line number and the original line will be eliminated.

Note 2. In the program mode, if keys are input when the cursor is not displayed, their corresponding characters are usually displayed on the leftmost column of the top line of the display unit. However, if the [►] or [◄] key is pressed when the prompt symbol is displayed, the contents of successive key inputs are displayed starting from the prompt position.

Note 3. When modifying a program, add the new command before deleting the old command. For example, to change the PRINT command in "30 PRINT 123" to some other command (i.e., LPRINT), first add the new command before deleting "PRINT".
If the PRINT command is deleted first, line number "30" and the number "123" combine, and the next line may be displayed or only the flashing cursor may be displayed.

Example 3—Using Variables in Programming

If you are unfamiliar with the use of numeric and string variables in BASIC, reread these sections in Chapter 4.

Using variables in programming allows much more sophisticated use of the PC-1350's computing abilities.

Remember, you assign simple numeric variables using any letter from A to Z:

```
A=5
```

To assign string variables you also use a letter, followed by a dollar sign. Do not use the same letter in designating a numeric and a string variable. You cannot designate A and A$ in the same program.

Remember that simple string variables cannot exceed 7 characters in length:

```
A$="TOTAL"
```

Note 1:
If an error occurs during program execution and if the location where it occurred was at the end of a line, press the ► key before inserting text here.

(example)

	ERROR 1 IN 20
CLS MODE ↑	20:S=π*R*_ ← Cursor display
► R ENTER	20:S=π*R*R

If the ► key was not pressed, the displayed program may disappear from the screen when keying in text to be added. Pressing the ENTER key displays the program added with the entered text.

Note 2:
The error or break message during program execution is displayed on the line following the display by manual operation or PRINT command. However, the error or break message may not be displayed for errors or breaks (execution of the STOP command or temporary interruption by pressing the BRK key) occurred when the display start position has been specified using the CURSOR command. The unit is, however, in the error or break mode. In the error mode, the alphabet and number keys do not function. Clear the error by pressing the CLS key.

The values assigned to a variable can change during the execution of a program, taking on the values typed in or computed during the program. One way to assign a variable is to use the INPUT verb. In the following program the value of A$ will change in response to the data typed in answering the inquiry "WORD?". Enter this program:

```
10 INPUT "WORD?";A$
20 B=LEN (A$)
30 PRINT "WORD␣IS␣";B;"␣LETTERS"
40 END
```

␣ means space

Since line 30 of this program is longer than 24 columns, the remaining part is displayed in the next line.

The second new element in this program is the use of the END statement to signal the completion of a program. END tells the computer that the program is completed. It is always good programming practice to use an END statement.

As your programs get more complex you may wish to review them before you begin execution. To look at your program, use the LIST command. LIST, which can only be used in the PROgram mode, displays programs beginning with the lowest line number.

Try listing this program:

Input	Display
LIST	**10: INPUT "WORD?"; A$** **20: B= LEN (A$)** **30: PRINT "WORD IS ";B;"** **LETTERS"**

Use the ↑ and ↓ arrows to move through your program until you have reviewed the entire program. To review a line which contains more characters than can be seen at one time, move the cursor to the extreme right of the display and the additional characters will appear on the screen. After checking your program, run it:

Input	Display
RUN	RUN WORD?_
HELP	RUN WORD? HELP_
ENTER	RUN WORD? HELP WORD IS 4. LETTERS

This is the end of your program. Of course you may begin it again by entering RUN. However, this program would be a bit more entertaining if it presented more than one opportunity for input. We will now modify the program so it will keep running without entering RUN after each answer.

Return to the PRO mode and use the up or down arrow (or LIST) to reach line 40. Press the up or down arrow key until the Line 40 comes to the top of the screen. (or type LIST 40 and Press ENTER)

Display

```
40:END
```

You may type 40 to Delete the entire line or use the ► to position the cursor over the E in End. Change line 40 so that it reads:

```
40: GOTO 10
```

Now RUN the modified program.

The GOTO statement causes the program to loop (keep repeating the same operation). Since you put no limit on the loop it will keep going forever (an "infinite" loop). To stop this program hit the BREAK (BRK) key.

When you have stopped a program using the BRK key, you can restart it using the CONT command. CONT stands for CONTinue. With the CONT command the program will restart on the line which was being executed when the BRK key was pressed.

Example 4—More Complex Programming

The following program computes N Factorial (N!). The program begins with 1 and computes N! up to the limit which you enter. Enter this program.

```
100 F=1: WAIT 118
110 INPUT "LIMIT?"; L
120 FOR N=1 TO L
130 F=F*N
140 PRINT N, F
150 NEXT N
160 END
```

Several new features are contained in this program. The WAIT verb in line 100 controls the length of time that displays are held before the program continues. The numbers and their factorials are displayed as they are computed. The time they appear on the display is set by the WAIT statement to approximately 2 seconds, instead of waiting for you to press ENTER .

Also in line 100, notice that there are two statements on the same line separated by a colon(:). You may put as many statements as you wish on one line, separating each by a colon, up to the 80 character maximum including [ENTER]. Multiple statement lines can make a program hard to read and modify, however, so it is good programming practice to use them only where the statements are very simple or there is some special reason to want the statements on one line.

Also in this program we have used the FOR verb in line 120 and the NEXT verb in line 150 to create a loop. In Example 3 you created an "infinite" loop which kept repeating the statements inside the loop until you pressed the [BRK] key. With this FOR/NEXT loop the PC-1350 adds 1 to N each time execution reaches the NEXT verb. It then tests to see if N is larger than the limit L. If N is less than or equal to L, execution returns to the top of the loop and the statements are executed again. If N is greater than L, execution continues with line 160 and the program stops.

You may use any numeric variable in a FOR/NEXT loop. You also do not have to start counting at 1 and you can add any amount at each step. See Chapter 9 for details.

We have labelled this program with line numbers starting with 100. Labelling programs with different line numbers allows you to have several programs in memory at one time. To RUN this program instead of the one at line 10 enter:

```
RUN 100
```

In addition to executing different programs by giving their starting line number, you can give programs a letter name and start them with the DEF key (see Chapter 6).

Storing Programs in the PC-1350's Memory

You will remember that settings, ReSerVe keys, and functions remain in the computer even after it is turned OFF. Programs also remain in memory when you turn off the PC-1350, or it undergoes an AUTO OFF. Even if you use the [BRK], CLear or CA keys the programs will remain.

Programs are lost from memory only when you perform the following actions:

* You enter NEW before beginning programming.
* You initialize the computer using the ALL RESET button.
* You create a new program using the SAME LINE NUMBERS as a program already in memory.
* You change the batteries.

This brief introduction to programming on the PC-1350 should serve to illustrate the exciting programming possibilities of your new computer. For more practice in programming exercises, please see Program examples.

Graphic Functions

The display unit (screen) of the PC-1350 is composed on 150 horizontal and 32 vertical dots (points). Simple pictures can be drawn on the screen using these dots. The following 6 commands are available for drawing pictures,

GPRINT: Graphic PRINT. This command is used to produce patterns with 8 vertical dots per unit.

GCURSOR: Graphic CURSOR. This command is used to specify the position when drawing a pattern with GPRINT.

PSET: Point SET. This command is used to light up or reverse a specified point (dot).

PRESET: Point RESET. This command clears the specified point (dot).

LINE: This command is used to draw a line or a square between 2 specified points.

POINT: This command is used to determine whether the specified dot is lit or not.

The basics of picture drawing will be described here. For details on the function of each command, see the description for each conmand.

Basically, there are 2 ways to draw pictures.

The first is to draw pictures by combining predetermined patterns. The other is to draw pictures by specifying and lighting up each dot (point) as necessary.

To display a combination of characters and numeric values along with graphics, use the graphic command after displaying the characters and numeric values.

(1) Drawing a Picture Using Predetermined Patterns

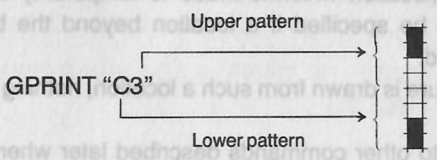
A picture is drawn by combining the 16 available patterns shown in the table below and using the GPRINT command.

Hexadecimal character	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Pattern																

The patterns in this table all use 4 dots (points).

However, in the GPRINT command, two patterns are combined vertically for an 8-dot pattern as shown in the example below. A picture is drawn by combining and lining up a number of 8-dot patterns.

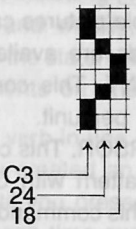
(Example)



A pair of hexadecimal numbers are used with the first specifying the lower pattern and the second specifying the upper pattern.

(Example)

```
GPRINT "C32418"
```

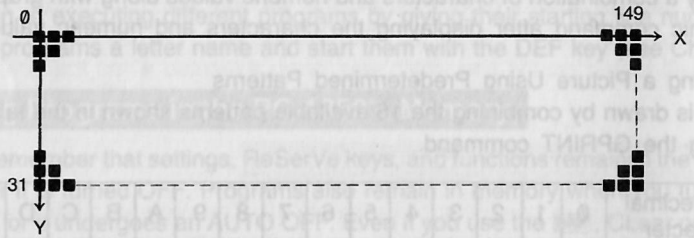


The 8-dot patterns specified by the hexadecimal number pairs are arranged to draw the picture.

(2) Specifying a Location on the Screen

As described, patterns are specified using the GPRINT command. The location where the pattern is to be displayed can be specified using the GCURSOR command. The display unit (graphic screen) of the PC-1350 is composed of 150×32 dots (points).

Each dot, like a point on an X-Y coordinate system, can be specified in the form of (x, y), where "x" is the horizontal direction and "y" is the vertical direction.



Note, however, that in normal X-Y coordinate systems, the point is higher with larger values of "y," while in the PC-1350, the point is lower on the screen.

The coordinates of the dots on the screen range 0–149 for "x" and 0–31 for "y."

Note: The values for "x" and "y" can be specified in the range of −32768 to +32767. A virtual location (location which is made to temporarily exist but does not actually exist) will be specified if a location beyond the boundaries of the screen is specified.

Therefore, if a picture is drawn from such a location, nothing may be displayed on the screen.

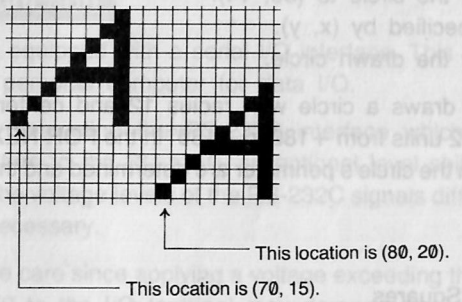
This also applies to other commands described later where the coordinates are specified with such as PSET, PRESET, and LINE commands.

The GPRINT command draws pictures by lining up groups of 8-dot patterns. If the location (dot) where the picture is to be displayed was specified with the GCURSOR command, the first 8-dot pattern is displayed on the 8 dots above and including the dot at the specified location. The rest of the pattern is drawn in sequence.

(Example)

```
 5 CLS
10 AA$="80402010181412FF"
20 GCURSOR (70, 15)
30 GPRINT AA$
40 GCURSOR (80, 20)
50 GPRINT AA$
```

* Executing the program will display the following pattern near the center of the screen.



(3) Drawing a Picture by Specifying Dots One-by-One

A picture is drawn by lighting up or clearing the specified dot using the PSET or PRESET command.

Each dot is specified in the same way as in the GCURSOR command.

(Example)

PSET (75, 15)	Lights up the dot at (75, 15)
PSET (75, 15), X	Lights up the dot at (75, 15) if cleared, and clears it if lit. (Specifying "X" reverses the dot.)
PRESET (75, 15)	Clears the dot at (75, 15).

(Example)

```
100 WAIT 0 : DEGREE
110 FOR A=180 TO -180
    STEP -2
120 B=SIN A*12          } *1
130 C=COS A*12          }
140 X=B+80              } *2
150 Y=C+14              }
160 PSET (X, Y)
170 NEXT A              <-*3
180 WAIT : GPRINT       <-*4
```

*1 Determines the coordinates of the perimeter of a circle with radius 12. The center is at (0, 0).
*2 Moves the center of the circle to (80, 14).
*3 Lights up the dot specified by (x, y).
*4 Continues to display the drawn circle.

Executing this program draws a circle with radius 12 and center at (80, 14). The angle is changed in 2 units from +180 to −180° in the FOR-NEXT loop. At each angle, the coordinates on the circle's perimeter are determined and the corresponding dot is lit.

(4) Drawing Lines and Squares

Although lines and squares can also be drawn using the PSET command, they can be easily drawn simply by specifying 2 dots in the LINE command.

(Example)

```
LINE (0, 0)-(149, 31)
```

Draws a line from (0, 0) to (149, 31).

```
LINE (30, 0)-(80, 31), B
```

Draws a square with its diagonal from (30, 0) to (80, 31). If B is changed to BF, the inside of the square will be filled.

```
LINE (30, 0)-(80, 31), X, BF
```

A filled box is drawn. However, if a dot within the square is already lit, it is cleared.
(Specifying "X" reverses the dots.)

(Example)

```
200 "A" :WAIT 0
210 LINE (60, 0)-(100, 31)
    , X, BF
220 GOTO 210
```


Executing this program draws a square with its diagonal from (60, 0) to (100, 31) and fills and clears it. With the picture previously drawn using the GPRINT and PSET commands still displayed, try executing the program by pressing DEF and A. The picture within the square will reverse.

Note: the graphic screen is 6 dots wider on the left side than the usual character and numeric display.
The PC-1350 is designed to retain the drawn image unless cleared. Consequently, if the program ends or is stopped, the graphic image may remain on the left side or top of the screen.
(The display is cleared by Pressing CLS.)

Note: Characters or numerals keyed in when the program is stopped by the graphic command may not be displayed. First clear the display (screen) using the CLS key and then key in.

Serial I/O Function

The PC-1350 is equipped with a serial I/O interface. This function can be used to connect with a personal computer for data I/O.

Note 1: When connecting the RS-232C interface which is equipped on many computers to the PC-1350, an optional level shifter becomes necessary since the voltage levels of the RS-232C signals differ. A connecting cable is also necessary.

Note 2: Exercise care since applying a voltage exceeding the allowable range of the PC-1350 to the I/O terminal may damage the internal parts.

Basics on the Use of the Serial I/O Interface

The circuit of the serial I/O interface is usually closed. If closed, data from the serial I/O terminal cannot be sent and the received data cannot be read.
Therefore, it is first necessary to open the circuit. The circuit is opened using the OPEN command. (If already open, ERROR 8 results.)

Further, the conditions to perform data I/O with the computer connected to the PC-1350 must be satisfied. In other words, the conditions for the signals must be the same for both the PC-1350 and the connected personal computer. If the conditions are different, the signals (data) cannot be read correctly resulting in data errors. The OPEN command can be used to set and modify the I/O conditions.
After the conditions for both sides are satisfied and the circuit opened, the following commands are used to perform data or program I/O.

LPRINT, LLIST, SAVE, LOAD, PRINT #1, INPUT #1

After the end of the data or program I/O, the circuit of the serial I/O interface is closed. Although the CLOSE command is used to close the circuit, the circuit also closes when the program ends (such as when the END command is executed) or when the RUN command is executed.

When writing a program which uses the serial I/O interface, the circuit must be opened, the I/O operation performed, and then the circuit must be closed as described above.

When using the serial I/O commands

When several programs are stored within the PC-1350 with the MERGE command, a program cannot be sent through the serial I/O interface with the SAVE or LOAD command. This is because the MERGE command of the PC-1350 differs from the MERGE command of other personal computers.

Note: The PC-1350 is not designed to wait during the I/O commands to the serial I/O interface and is not equipped with a timer function to interrupt communication with the connected equipment.

Therefore, if the connected equipment is not ready to communicate (such as when the power is off) while commands are being executed or if communication at the connected equipment is interrupted, the PC-1350 cannot stop the executed commands and will continue execution.

If this is the case, press the [BRK] key and stop the execution.

CHAPTER 6
SHORTCUTS

The PC-1350 includes several features which make programming more convenient by reducing the number of keystrokes required to enter repetitive material.

One such feature is in the availability of abbreviations for verbs and commands (See Chapter 9).

This chapter discusses two additional features which can eliminate unnecessary typing—the DEF key and the Reserve mode.

The DEF Key and Labelled Programs

Often you will want to store several different programs in the PC-1350's memory at one time. (Remember that each must have unique line numbers). Normally, to start a program with a RUN or GOTO command, you need to remember the beginning line number of each program (see Chapter 9). But, there is an easier way! You can label each program with a letter and execute the program using only two keystrokes. This is how to label a program and execute it using DEF:

Note: Put a label on the first line of each program that you want to reference. The label consists of a single character in quotes, followed by a colon:

```
10: "A": PRINT "FIRST"
20: END
80: "B": PRINT "SECOND"
90: END
```

Any one of the following characters can be used: A, S, D, F, G, H, J, K, L, =, Z, X, C, V, B, N, M, and SPC. Notice that these are the keys in the last two rows enclosed with the white line to make it easier for you to remember.

Note: To execute the program, instead of typing RUN 80 or GOTO 10, you need only press the [DEF] key and then the letter used as a label. In the above example, pressing [DEF] and then 'B' would cause 'SECOND' to appear on the display.

When DEF is used to execute a program, variables and mode settings are affected in the same way as when GOTO is used. See Chapter 9 for details.

RESERVE Mode

Another timesaving feature of the PC-1350 is the ReSerVe mode.

Within the memory of the PC-1350, 144 characters are designated for "Reserve Memory". You can use this memory to store frequently used expressions, which are then recalled by a simple two keystroke operation.

Note:
- You store the strings in the RESERVE mode and recall them for use in the RUN and PROgram modes.
- The PC-1350 has a reserve memory of 144 bytes. Set up to 144 bytes, including the reserve key, in the reserve memory. A BASIC command, function, number, or a alphabetic character is 1 byte.
  Example: A: S: + − 1 2 A B SIN COS INPUT RUN.....
  (1 byte each)
- The length of a reserved string for one key is a maximum of up to 80 key commands, including the reserve key and [ENTER] key.

Try this example of storing and recalling a reserved string.

Change the PC-1350 into RESERVE mode by pressing the [SHIFT] and [MODE] keys. Notice that the mode indicator "RUN" and "PRO" disappear and the message "RESERVE MODE" is displayed when the reserve mode is set.

Type NEW followed by the [ENTER] key. This will clear out any previously stored characters in the same way NEW clears out stored programs in the PROgram mode.

Type [SHIFT] followed by 'A':

Input	Display
[SHIFT] A	**A: _**

Notice that the 'A' appears in the display at the left followed by a colon.

Enter the word 'PRINT' and press the [ENTER] key:

Input	Display
PRINT [ENTER]	**A: PRINT**

A space appears after the colon signalling you that 'PRINT' is now stored in the reserve memory under the letter A.

Switch the PC-1350 into PROgram mode. Type NEW followed by [ENTER] to clear the program memory. Type '10' as a line number and then press [SHIFT] and the 'A' key: the 'A' key:

Input	Display
10 [SHIFT] A	**10 PRINT _**
[ENTER]	**10: PRINT**

Immediately the word 'PRINT' will appear in the display after the line number.

Any character sequence can be stored in reserve Memory. The stored strings can be recalled at any time in either the PROgram or the RUN mode by typing [SHIFT] and the key that the string is stored under. The keys available are the same as those used with DEF, i.e., those in the area enclosed with the white line.

To edit a stored character sequence, switch into the reserve mode and press [SHIFT] and the key under which the sequence is stored. You can then edit using the Left Arrow, Right Arrow, DEL, and INS keys in the same way as in other modes.

When the last character in a stored sequence is a '@' character, it is interpreted as [ENTER] when the sequence is recalled. For example, if you store the string "GOTO 100@" under the 'G' key, typing [SHIFT] and 'G' in the RUN mode immediately starts execution of the program at line 100. Without the '@' character, you must press [ENTER] after the [SHIFT] and 'G' to begin execution.

To delete reserve programs:

1. As you know, NEW [ENTER] keys clear all reserve memories.
   Please note that the above key operation must be done in the RESERVE mode.
2. To delete a reserve memory, use the [SPC] or [DEL] key as described below:
   Example: Clear A*A which is reserved in the key number [S].

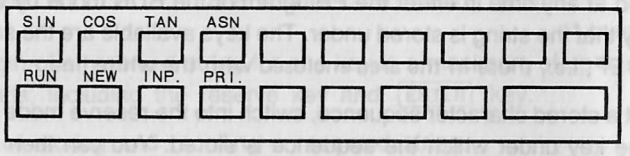
Input	Display	Remarks
[SHIFT] S	S:_	Reserve mode
A*A [ENTER]	S:A*A	
[CLS]	>	
[SHIFT] S	S:A*A	
[◄] or [►]	S:A*A	
[DEL] [DEL] [DEL]	S:_	Delete A*A
[ENTER]	>	

Templates

Two templates are provided with the PC-1350. You can use these templates to help you remember frequently used ReSerVe sequences or DEF key assignments. After you have labelled the programs or created the sequences, mark the templates so you know what is associated with each key. You can then execute programs or recall sequences using the two-keystroke operation.

For example, if you have one group of programs which you often use at the same time, label the programs with letters and mark the template so that you can easily begin execution of any of the programs with two keystrokes. You might also store frequently used BASIC commands and verbs in the Reserve Memory and mark a template to speed up entering BASIC programs:

Example:



CHAPTER 7
USING THE CE-126P PRINTER/CASSETTE INTERFACE

The optional CE-126P Printer/Cassette Interface allows you to add a printer and to connect a cassette recorder to your SHARP PC-1350 Computer.
The CE-126P features:

* 24 character wide thermal printer.
* Convenient paper feed and tear bar.
* Simultaneous printing of calculations as desired.
* Easy control of display or printer output in BASIC.
* Built-in cassette interface with remote function.
* Manual and program control of recorder for storing programs, data.
* Dry battery operation for portability.

For connecting the PC-1350 to the CE-126P, refer to the instruction manual which is supplied with the CE-126P.

Using the Printer

If you are using the PC-1350 for manual calculation, you may use the CE-126P to simultaneously print your calculations.

This is easily accomplished by pressing the [SHIFT] key and then the [ENTER] key (P ↔ NP) while in the RUN mode.
After this, when you press the [ENTER] at the end of a calculation, the contents of the display will be printed on one line and the results will be printed on the next. For example:

Input	Paper
300/50 [ENTER]	300/50 6.

You may print output on the printer from within BASIC programs by using the LPRINT statement (see Chapter 9 for details). LPRINT can be used in the same form as the PRINT statement.

Programs which have been written with PRINT can be converted to work with the printer by including a PRINT=LPRINT statement in the program (see Chapter 9 for details). All PRINT statements following this statement will act as if they were LPRINT statements. PRINT=PRINT will reset this condition to its normal state. This structure may also be included in a program in an IF statement allowing a choice of output at the time the program is used.

You may also list your programs on the printer with the LLIST command (see Chapter 9 for details). If used without line numbers LLIST will list all program lines currently in memory in their numerical order by line number. A line number range may also be given with LLIST to limit the lines which will be printed. When program lines are longer than 24 characters, two or more lines may be used to print one program line. The second and succeeding line will be indented four or six characters so that the line number will clearly identify each separate program line. (Line number, 1 to 999: four; over 999: six)

Caution:

- In case an error (ERROR code 8) occurs due to a paper misfeed, tear off the paper tape, and pull the remaining part of the paper tape completely out of the printer. Then press the [CLS] key to clear the error condition.
- When the printer is exposed to strong external electrical noise, it may print numbers at random. If this happens, depress the [BRK] key to stop the printing. Turn the CE-126P power off and on, and then press the [CLS] key. Pressing the [CLS] key will return the printer to its normal condition.

  When the printer causes a paper misfeed or is exposed to strong external electrical noise while printing, it may not operate normally. If this occurs, depress the [BRK] key to stop printing. (Release the paper misfeed.) Turn the CE-126P power off and on, and then press the [CLS] key.
- When the CE-126P is not in use, turn off the printer switch to save the battery life.

Using the Cassette Interface

Using this cassette interface will allow you to store programs and data from the computer onto cassette tape (of course you'll also need a cassette recorder such as we sell for this pocket computer system: model CE-152). Once on tape, you can load these programs and data back into the computer with a simple procedure.

Connecting the CE-126P to a Tape Recorder

Only three connections are necessary:

1. Connect red plug into the MICrophone jack on the cassette recorder.
2. Connect gray plug into the EARphone jack on the cassette recorder.
3. Connect the black plug into the REMote jack on the cassette recorder.

Cassette Tape Recorder

We recommend you to use an optional cassette tape recorder CE-152 for your pocket computer system. The CE-152 is designed to match the PC-1350 and records programs and data via the CE-126P cassette interface. Any recorded program can be retrieved and reloaded into the PC-1350.

When you use any other cassette tape recorder than the CE-152:

On the next page are described the minimum tape recorder specifications necessary for interfacing with the CE-126P:

Item	Requirements
1. Recorder Type	Any tape recorder, standard cassette or micro-cassette recorder, may be used in accordance with the requirements outlined below.
2. Input Jack	The recorder should have a mini-jack input labeled "MIC". Never use the "AUX" jack.
3. Input Impedance	The input jack should be a low impedance input (200~1,000 OHM.)
4. Minimum Input Level	Below 3 mV or −50 dB.
5. Output jack	Should be a minijack labeled "EXT. (EXTernal speaker)", "MONITOR", "EAR (EARphone)" or equivalent.
6. Output impedance	Should be below 10 OHM.
7. Output level	Should be above 1V (practical maximum output above 100 mW)
8. Distortion	Should be within 15% within a range of 2 KHz through 4 kHz.
9. Wow and Flutter	0.3% maximum (W.R.M.S.)
10. Other	Recorder motor should not fluctuate in speed.

* In case the miniplug provided with the CE-126P is not compatible with the input/output jacks of your tape recorder, special line conversion plugs are available on the market.

Note: Some tape recorders may not perform properly due to different specifications. Or those tape recorders having distortion, increased noise, and power deterioration after long years of use may not show satisfactory results owing to changes in their electrical characteristics.

Operating the Cassette Interface and Recorder

Recording (saving) onto magnetic tape

See Tape Notes.

1. Turn off the REMOTE switch on the CE-126P.
2. Enter a program or data into the Computer.
3. Load tape into the tape recorder.
   Determine the position on the tape where you want to record the program.
   - When using a tape, be sure the tape moves past the clear leader (non-magnetic mylar material).
   - When using a tape already partially recorded, search for a location where no recording exists.
4. Connect the Interface's red plug to the tape recorder's MIC jack and the black plug to the REM jack.
5. Turn on the REMOTE switch.
6. Simultaneously press record and play buttons on the tape recorder (to put it in record mode).
7. Enter recording instructions (CSAVE statement, PRINT # statement), and press the [ENTER] key for execution.

   First set the unit to "RUN" or "PRO" mode. Next push the following keys:
   [C] [S] [A] [V] [E] [SHIFT] ["] file name [SHIFT] ["] [ENTER].
   (To write the contents of data memory onto tape, use the following method:
   [P] [R] [I] [N] [T] [SHIFT] [#] [ENTER].)
   E.g., [C] [S] [A] [V] [E] [SHIFT] ["] [A] [A] [SHIFT] ["] [ENTER]

   When you press the [ENTER] key, tape motion will begin, leaving about an 8-second non-signal blank. (Beep tone is recorded.) After that, the file name and its contents are recorded.
8. When the recording is complete, the PROMPT symbol (>) will be displayed and the tape recorder will automatically stop. Now you have your program on tape (it still is in the Pocket Computer also).
   When data is to be automatically recorded by program execution (PRINT # statement, not manual operation), set up steps 1 thru 6 before executing the program.

   To aid you in locating programs on tapes, use the tape counter on the recorder.

Collating the Computer and Tape Contents

See tape Notes.

After loading or transferring a program to or from tape, you can verify that the program on tape and program in the Pocket Computer are identical (and thus be sure that everything is OK before continuing your programming or execution of programs).

1. Turn off the REMOTE switch.
2. With cassette in the recorder, operate the tape motion controls to position tape at the point just before the appropriate file name to be checked.
3. Connect gray plug to EARphone and black plug to REMote jacks.
4. Turn on the REMOTE switch.
5. Press PLAY button of recorder.
6. Input a CLOAD? statement and start execution with [ENTER] key. Do this as follows: Set unit to "RUN" or "PRO" mode. Enter the following key sequence—

The file name which you used previously.

[C] [L] [O] [A] [D] [SHIFT] [?] [SHIFT] ["] [A] [A] [SHIFT] ["] [ENTER]

The Pocket Computer will automatically search for the specified file name and will compare the contents on tape with the contents in memory.

During the collation, the mark "*" is shown at the rightmost digit of the bottom line of the display. The mark "*" will disappear when the collation is completed. While a file name is being retrieved, no "*" mark will be displayed as the collation is not started yet.
(The same occurs when the first program is read without a file name.)

If the programs are verified as being identical, a PROMPT symbol (>) will be displayed on the Pocket Computer.

If the programs differ, execution will be interrupted and an Error code 8 will be displayed. If this occurs, try again.

Loading from a magnetic tape

See Tape Notes.

To load, transfer, or read out programs and data from magnetic tape into the Pocket Computer, use the following procedure.

1. Turn off the REMOTE switch.
2. Load tape in the tape recorder. Position tape just before the portion to be read out.

3. Connect the gray plug to the EAR jack on the tape recorder, and the black plug to the REM jack.
   [In using a tape recorder having no REM terminal, press the PAUSE button to make a temporary stop.]
4. Turn on the REMOTE switch.
5. Push the PLAY button on the tape recorder (to put unit in playback mode).

   Set the VOLUME control to middle or maximum.
   Set Tone to maximum treble.
6. Input transfer instructions (CLOAD statement, INPUT # statement), and press [ENTER] key for execution.

   Put the unit into "RUN" or "PRO" mode. Then push the following keys:
   [C] [L] [O] [A] [D] [SHIFT] ["] file name [SHIFT] ["] [ENTER] .
   (To load the contents of the data memory, push as follows: [I] [N] [P] [U] [T] [SHIFT] [#] [ENTER].)

   E.g., [C] [L] [O] [A] [D] [SHIFT] ["] [A] [A] [SHIFT] ["] [ENTER]

   The specified file name will be automatically searched for and its contents will be transferred into the Pocket Computer.
   The mark "*" appears while loading the designated CSAVEd program from the tape to the computer's memory.
   (The same occurs when the first program is read without a file name,) The mark "*" disappears when the load is performed completely.
7. When the program has been transfered the computer will automatically stop the tape motion and display the PROMPT (>) symbol.

   To transfer data (INPUT # statement) in the course of a program, set up steps 1 thru 5 prior to executing the program.

Notes:
- If an error occurs (error code "8" is displayed), start over from the beginning. If the error continues, adjust volume up or down slightly.
- If the error code is not displayed but tape motion continues transferring is improper. Press [ON BRK] key (to "break") to stop the tape. Repeat steps.
- If the error remains or the tape continues to run after several attempts to correct the problem, try cleaning and demagnetizing the Recorder's tape head.

Tapenotes

1) For any transfer or collation, use the tape recorder that was used for recording. If the tape recorder for transfer or collation is different from that used for recording, transfer or collation may not be possible.
2) Always use only the highest quality tape for programs and data storage (economy grade audio type tape may not provide the proper characterstics for digital recordings).
3) Keep the tape heads and tape handling parts clean—use a cassette cleaner tape to keep everything clean.
4) Volume setting—set to middle or maximum level.
   Volume level can be very important when reading in data from the recorder; make slight adjustments as required to obtain error-free data transfer. A slight adjustment either up or down may result in perfect recordings every time.
5) Be sure all connections between the pocket computer and cassette interface are secure. And be sure the connections between interface and recorder are secure and dirt-free.
6) If problems occur when using AC power for the CE-126P and/or the recorder, use battery power instead (sometimes the AC power connection also adds some "hum" to the signal which upsets proper digital recordings).
   - To connect the AC adaptor to the CE-126P, turn the CE-126P power off and then connect the adaptor to the CE-126P.
7) Tone control—set to maximum treble.
8) When recording programs or data on the used tape, erase the portion before writing and execute the recording command. (Make sure that the previous program is completely erased without any portion remaining.)

CHAPTER 8
USING THE RAM CARD (OPTION)

The PC-1350 is capable of using optional RAM cards CE-201M or CE-202M.

The CE-201M and CE-202M are RAM (Random Access Memory) cards with capacities of 8 K bytes and 16 K bytes, receptively.

The program/data area of the PC-1350 can be expanded by mounting the RAM card. Further, a program can be saved even if the RAM card is removed after it has been stored with the program. Therefore a number of RAM cards can each be stored with a different program so that different programs can be executed simply by changing the RAM card.

1. Mounting the RAM Card

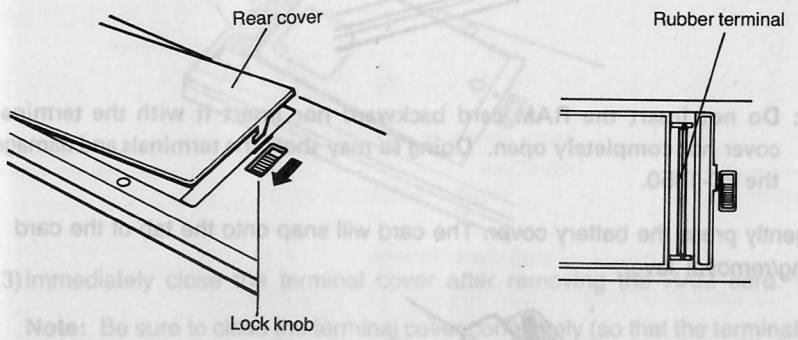
If the RAM card is to be used for the first time, be sure to insert the battery into the RAM card.

(1) Turn off the PC-1350.

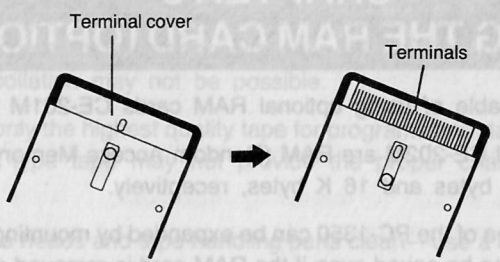
Be sure turn off the PC-1350 before installing or removing the RAM card.

(2) Slide the rear cover lock in the direction of the arrow shown in the figure and remove the rear cover.

Note: Do not touch the circuits, components, rubber terminals, etc., found within the PC-1350.

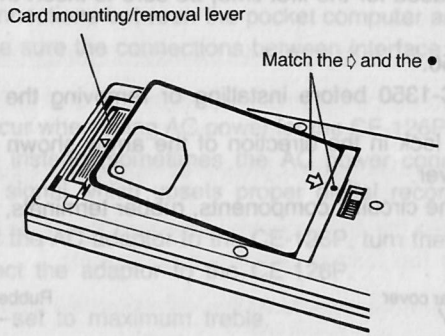


(3) Open the Terminal cover of the RAM card.



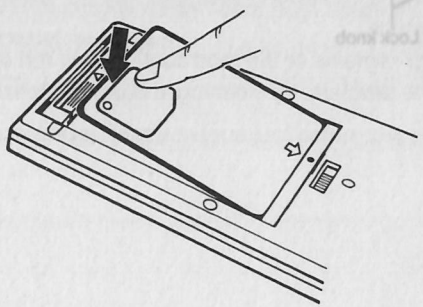
Note: Completely open the terminal cover (so that it does not cover the terminals). Also, do not touch the terminals.

(4) Insert the terminal end of the RAM card down into the PC-1350 as shown in the figure.

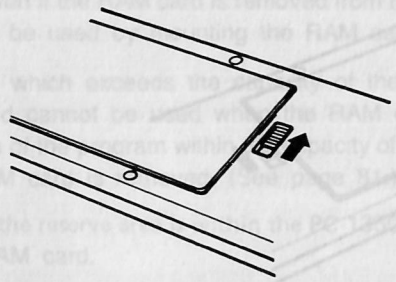


Note: Do not insert the RAM card backward nor insert it with the terminal cover not completely open. Doing so may short the terminals and damage the PC-1350.

Then, gently press the battery cover. The card will snap onto the tab of the card mounting/removal lever.



(5) Attach the rear cover and slide the lock to the "LOCK" position.

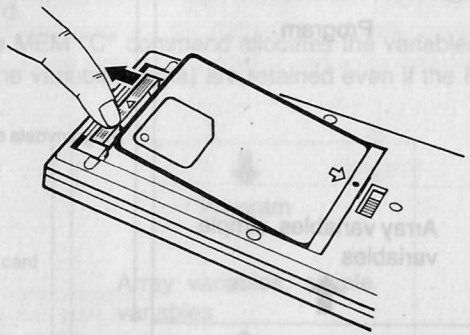


Note: Be sure to slide the lock to the "LOCK" position. The PC-1350 will not operate if the lock is not at the "LOCK" position even though the batteries are inserted. If the power is turned on with the lock unlocked, be sure to lock it and then turn the power off and then on again.

2. Removing the RAM Card

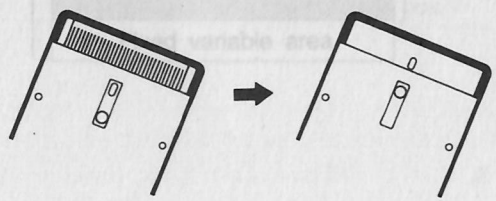
(1) Turn the PC-1350 off.

(2) Remove the rear cover of the PC-1350 as described in 1. above. Pull the card mounting/removal lever in the direction of the arrow as shown in the figure. This releases the RAM card from the tab so that it can be removed.

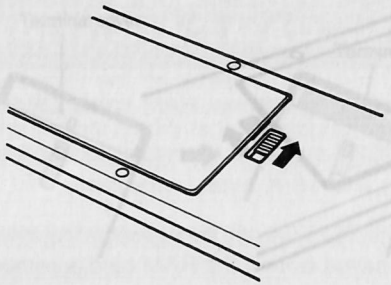


(3) Immediately close the terminal cover after removing the RAM card.

Note: Be sure to close the terminal cover completely (so that the terminals cannot be seen).



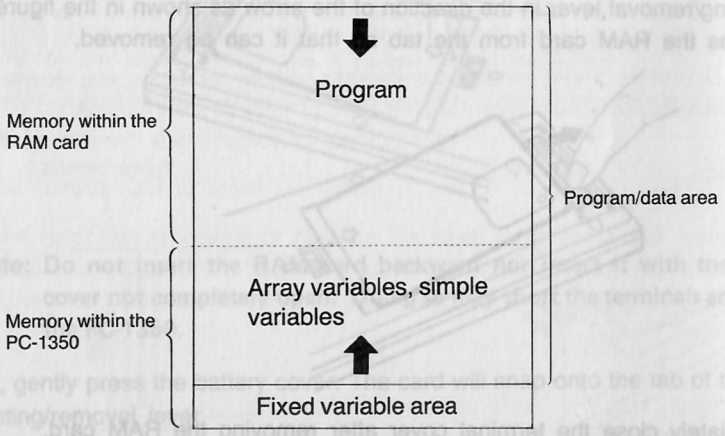
(4) Attach the rear cover to the PC-1350 and slide the lock to the "LOCK" position.



3. Using the RAM Card

When the RAM card, with nothing stored, is mounted in the PC-1350, the memory within the PC-1350 and the memory within the RAM card are linked so that the program/data area becomes a single memory area.

As shown in the figure below, the program is written on the memory within the RAM card while array variables and simple variables are allocated in the memory within the PC-1350.



The program written within the RAM card is retained (by the battery inserted into the RAM card) even if the RAM card is removed from the PC-1350. Therefore, the same program can be used by mounting the RAM card again into the PC-1350.

If a program which exceeds the capacity of the RAM card is written, it will be destroyed and cannot be used when the RAM card is removed.
Keep the size of the program within the capacity of the RAM card to preserve it even after the RAM card is removed. (See page 81.)

Note: Since the reserve area is within the PC-1350, its contents cannot be stored in the RAM card.

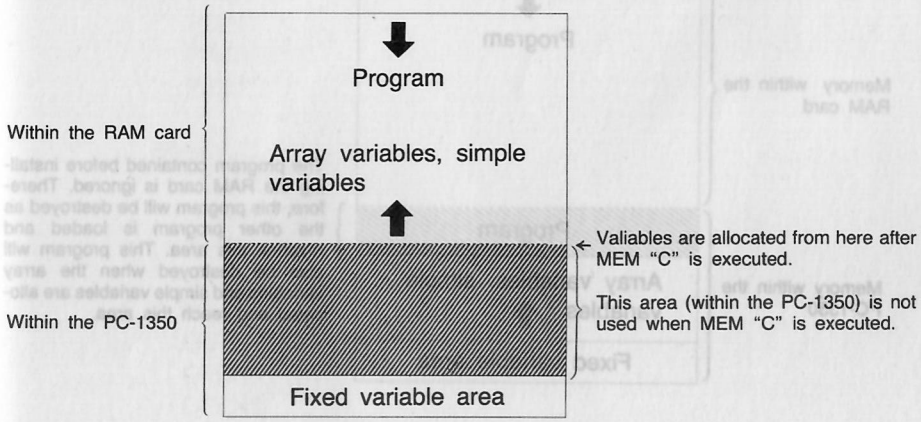
Capacity of the Program/Data Area (bytes)

PC-1350+CE-201M	within CE-201M only	PC-1350+CE-202M	within CE-202M only
11262	8142	19454	16334

4. Saving Data within the RAM Card

As shown in the previous figure, variables (such as array variables and simple variables) are allocated in the memory within the PC-1350. Therefore, if the RAM card is removed from the PC-1350, the variables will be cleared. The following command, MEM "C", can be specified so that the variables are allocated in the memory within the RAM card.

Executing the MEM "C" command allocates the variables in the memory within the RAM card. The variables (data) are retained even if the RAM card is removed from the PC-1350.


Program
Within the RAM card
Array variables, simple variables
Valiables are allocated from here after MEM "C" is executed.
This area (within the PC-1350) is not used when MEM "C" is executed.
Within the PC-1350
Fixed variable area

* The setting due to the MEM "C" command is cleared by the MEM "B" command. If the MEM "C" command is executed, however, the previously allocated array variables and simple variables will be cleared.

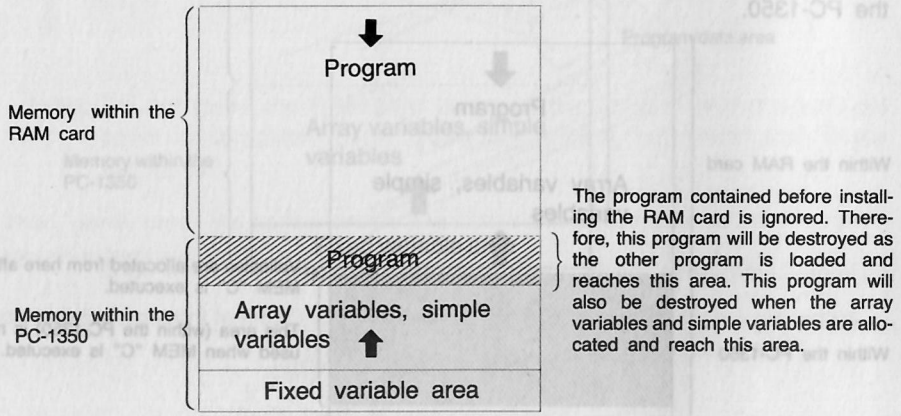
Note: The setting for MEM "C" is cleared when the program which is written exceeds the capacity of the RAM card. The setting changes to MEM "B".

The setting for the MEM "C" command is stored within the RAM card. Removing the RAM card from the PC-1350 invalidates the setting. Installing the RAM card once again restores the setting.

* To keep the data stored within the RAM card, define a key with a program, and start the program using the defined key. When the program is executed using the RUN command, the array variables and simple variables will be erased. Therefore, do not use the RUN command.
* Fixed variables cannot be saved on the RAM card.
  Therefore, to save data, use array variables or simple variables.

5. Precautions When Using the RAM Card

* If the memory within the PC-1350 contains a program when the RAM card is mounted, the program will be ignored. Therefore, the program already stored in the memory of the PC-1350 cannot be executed (nor called). If a program which exceeds the capacity of the RAM card is written, the program stored in the memory of the PC-1350 will be destroyed. The program may also be destroyed if variables are allocated in the memory with the PC-1350. (If the program has not been destroyed, it can be executed after the RAM card is removed.)


Program
Memory within the RAM card
Program
Memory within the PC-1350
Array variables, simple variables
Fixed variable area
The program contained before installing the RAM card is ignored. Therefore, this program will be destroyed as the other program is loaded and reaches this area. This program will also be destroyed when the array variables and simple variables are allocated and reach this area.

* The stored program and data are cleared when the battery in the RAM card is replaced (or removed). Therefore, if a valuable program is stored in the RAM card, it is recommended that the program be recorded beforehand on tape.
* If the battery in the RAM card is replaced while the RAM card is installed in the PC-1350, the contents are not cleared.
* When the new battery is replaced. the contents of the RAM card can be retained for approx. 34 months for the CE-201M and 18 months for the CE-202M. (When removed from the PC-1350).

Note: Do not press the all reset switch on the rear of the PC-1350 while the RAM card is installed.
The program and data stored within the RAM card are cleared if the all reset switch is carelessly pressed. If it becomes necessary to press the all reset switch, refer to page 14, and press the all reset switch while holding down the appropriate key.

CHAPTER 9
BASIC REFERENCE

The following chapter is divided into three sections:

Commands: Instructions which are used outside a program to change the working environment, perform utilities, or control programs.

Verbs: Action words used in programs to construct BASIC statements.

Functions: Special operators used in BASIC programs to change one variable into another.

However, commands related to graphic, serial I/O, and text functions are summarized in the following pages.

Graphic ..... pages 166 ~ 178
Serial I/O ..... pages 179 ~ 193
Text function ..... pages 194 ~ 196

Commands and verbs are arranged alphabetically. Each entry is on a separate page for easy reference. The contents of each section are shown in the tables below so that you can quickly identify the category to which an operator belongs. Functions are grouped according to four categories and arranged alphabetically within each category.

Note: Commands, verbs, and functions must be typed in the upper case character mode (normal mode).

Commands

Program Control
- CONT
- GOTO*
- NEW
- RUN

Cassette Control
- CLOAD
- CLOAD?
- CSAVE
- INPUT #*
- MERGE
- PRINT #*

Debugging
- LIST
- LLIST
- TROFF*
- TRON*

Graphic
- GCURSOR*
- GPRINT*
- LINE*
- POINT
- PRESET*
- PSET*

Serial I/O
- CLOSE*
- CONSOLE*
- INPUT#1*
- LLIST
- LOAD
- LPRINT*
- OPEN*
- OPEN$
- PRINT#1*
- SAVE

Variables Control
- CLEAR*
- MEM*
- DIM*

Angle Mode Control
- DEGREE*
- GRAD*
- RADIAN*

Other
- BEEP*
- PASS
- RANDOM*
- USING*
- WAIT*

Text
- BASIC
- TEXT

* These commands are also BASIC verbs. Their effect as commands is identical to their effect as verbs so they are not described in the command reference section. See the verb reference section for more information.

Verbs

Control and Branching

- CHAIN
- END
- FOR...TO...STEP
- GOSUB
- GOTO
- IF...THEN
- NEXT
- ON...GOSUB
- ON...GOTO
- RETURN
- STOP

Assignment and Declaration

- CLEAR
- DIM
- LET

In PUT and Output

- AREAD
- CSAVE
- CURSOR
- DATA
- GCURSOR
- GPRINT
- INPUT
- INPUT#
- INPUT#1
- LINE
- LOAD
- LPRINT
- PAUSE
- PRESET
- PRINT
- PRINT#
- PRINT#1
- PSET
- READ
- RESTORE
- USING
- WAIT

Other

- BEEP
- CLOSE
- CLS
- CONSOLE
- DEGREE
- GRAD
- OPEN
- RADIAN
- RANDOM
- REM
- TROFF
- TRON

Functions

Pseudovariables

- INKEY$
- MEM
- MEM$
- OPEN$
- PI
- POINT

String Functions

- ASC
- CHR$
- LEFT$
- LEN
- MID$
- RIGHT$
- STR$
- VAL

Numeric Functions

- ABS
- ACS
- ASN
- ATN
- COS
- DEG
- DMS
- EXP
- INT
- LN
- LOG
- RND
- SGN
- SIN
- SQR
- TAN

COMMANDS

1 **CLOAD**
2 **CLOAD** "filename"

Abbreviations: CLO., CLOA.

See also: CLOAD?, CSAVE, MERGE, PASS

Purpose

The CLOAD command is used to load a program saved on cassette tape. It can only be used with the optional CE-126P and CE-152.

Use

The first form of the CLOAD command clears the memory of existing programs and loads the first program stored on the tape, starting at the current position.

The second form of the CLOAD command clears the memory, searches the tape for the program whose name is given by "filename", and loads the program.

If the PC-1350 is in PROgram or RUN mode, program memory is loaded from the tape. When the PC-1350 is in the reserve mode, reserve memory is loaded. Care should be taken not to load programs into reserve memory or reserve characters into program memory.

Examples

CLOAD Loads the first program from the tape.
CLOAD "PRO3" Searches the tape for the program named 'PRO3' and loads it.

Notes:
1. The computer cannot identify the stored contents as a program or a reserve. Therefore, if a mode is designated incorrectly, the reserved contents may be transferred to the program area or the program to the reserve area, causing all computer housekeeping to remain inoperative. If this happens, reset the computer by pressing the RESET button on the back of the computer.
2. If the designated file name is not retrieved, the computer will continue to search the file name even after the tape reaches the end. In this case, stop the retrieval function by pressing the ON BRK key. This applies to MERGE, CHAIN, CLOAD? and INPUT # commands to be described later.
3. If an error occurs during CLOAD or CHAIN command (to be described later) execution, the program stored in the computer will be invalid.

- During the loading the mark "*" is shown at the right most digit of the bottom line of the display. The mark "*" will disappear when the loading is completed. While a file name is being retrieved, no "*" mark will be displayed as the loading has not started yet.
  (The same occurs when the first program is read without a file name.)

Note: The "*" mark usually disappears when the transfer ends. However, if a command such as CLOAD is executed at the 4th line of the display, the "*" mark does not disappear even if the transfer ends but moves to the 3rd line. The prompt symbol ">" appears when the transfer ends.

1 **CLOAD?**
2 **CLOAD?** "filename"

Abbreviations: CLO.?,CLOA.?

See also: CLOAD, CSAVE, MERGE, PASS

Purpose

The CLOAD? command is used to compare a program saved on cassette tape with one stored in memory. It can only be used with the optional CE-126P and CE-152.

Use

To verify that a program was saved correctly, rewind the cassette tape to the beginning of the program and use the CLOAD? command.

The first form of the CLOAD? command compares the program stored in memory with the first program stored on the tape, starting at the current position.

The second form of the CLOAD? command searches the tape for the program whose name is given by "filename" and then compares it to the program stored in memory.

Examples

CLOAD? Compares the first program from the tape with the one in memory.

CLOAD? "PRO3" Searches the tape for the program names 'PRO3" and compares it to the one stored in memory.

* The "*" mark is displayed at the bottom right of the display unit when the program is being verified. At the end of verificatin, the "*" mark disappears and the prompt is displayed.
  However, if the CLOAD? command is executed on the 4th line of the display, the "*" mark does not disappear but moves to the 3rd line.

1 **CONT**

Abbreviations: C.,CO.,CON.

See also: RUN, STOP verb

Purpose

The CONT command is used to continue a program which has been temporarily halted.

Use

When the STOP verb is used to halt a program during execution, the program can be continued by entering CONT in response to the prompt.

When a program is halted using the BRK key, the program can be continued by entering CONT in response to the prompt.

CONT also functions when the program is temporarily interrupted due to command such as PRINT and GPRINT.

Examples

CONT Continues an interrupted program execution.

1 **CSAVE**
2 **CSAVE** "filename"
3 **CSAVE**, "password"
4 **CSAVE** "filename", "password"

Abbreviations: CS., CSA., CSAV.

See also: CLOAD, CLOAD?, MERGE, PASS.

Purpose

The CSAVE command is used to save a program to cassette tape. It can only be used with the optional CE-126P and CE-152.

Use

The first form of the CSAVE command writes all of the programs in memory on to the cassette tape without a specified file name.

The second form of the CSAVE command writes all of the programs in memory on to the cassette tape and assigns the indicated file name.

The third form of the CSAVE command writes all of the programs in memory on to the cassette tape without a specified file name and assigns the indicated password. Programs saved with a password may be loaded by anyone, but only someone who knows the password can list or modify the programs. (See discussion under PASS command).

The fourth form of the CSAVE command writes all of the programs in memory on to the cassette tape and assigns them the indicated file name and password.

If the PC-1350 is in PROgram or RUN mode, program memory is loaded to the tape. When the PC-1350 is in the Reserve mode, reserve memory is loaded.

Examples

CSAVE "PRO3", "SECRET" Saves the programs now in memory on to the tape under the name 'PRO3', protected with the password 'SECRET'.

1 **GOTO** expression

Abbreviations: G., GO., GOT.

See also: RUN

Purpose

The GOTO command is used to start execution of a program.

Use

The GOTO command can be used in place of the RUN command to start program execution at the line number specified by the expression.

GOTO differs from RUN in eight respects:

1) The value of the interval for WAIT is not reset.
2) The display format established by USING statements is not cleared.
3) Variables and arrays are preserved.
4) PRINT=LPRINT status is not reset.
5) The pointer for READ is not reset.
6) The cursor specification is maintained.
7) The horizontal direction of the graphic cursor is specified with 0. The setting for the vertical direction is maintained.
8) The serial I/O circuit is not closed.

Execution of a program with GOTO is identical to execution with the DEF key.

Examples

GOTO 100 Begins execution of the program at line 100.

1 **LIST**
2 **LIST** line number
3 **LIST** "label"

Abbreviations: L., LI., LIS.

See also: LLIST

Purpose

The LIST command is used to display a program.

Use

The LIST command may only be used in the PROgram mode.

* With format (1), the program is displayed from its first line until the display is full.
* With format (2), the program is displayed from the line of the specified line number until the display is full.
  If the line for the specified number does not exist, the program will be displayed from the line with the next largest number which does exist.
* With format (3), the program is displayed from the line written with the specified label until the display is full.
* When programs are merged with the MERGE command, the LIST command functions for the last program.
  However, if the label specified in format (3) does not exist in the last program, it is searched for in sequence from the first program. If the specified label is found, the line containing it is displayed. If a password has been set the LIST command is ignored.

Examples

LIST 100 Displays line number 100.

1 **LLIST**
2 **LLIST** expression
3 **LLIST** expression 1, expression 2
4 **LLIST** expression,
5 **LLIST**, expression

Abbreviations: LL., LLI., LLIS.

See also: LIST

Purpose

The LLIST command is used for printing a program on the optional CE-126P.

Use

* When the serial I/O interface is open due to the OPEN command, the LLIST command outputs the program at the serial I/O interface terminal. (See page 182.)
  To return the program printing command to the printer, execute the CLOSE command.

The LLIST command may be used in the PROgram or RUN mode.

The first form prints all of the programs in memory.

The second form prints only the program line whose line number is given by expression.

The third form prints the statements from the line number with the nearest line equal to or greater than the value of expression 1 to the nearest line equal to or greater than the value of expression 2. There must be at least two lines between the two numbers.

The forth form prints all program lines beginning with the line whose number is given by the expression.

The fifth form prints all program lines up to, and including, the line whose number is given by the expression.

* When programs are merged with the MERGE command, the LLIST command functions for the last program. To list a program stored earlier, execute.

  LLIST "label",

If a password has been set the LLIST command is ignored.

Examples

LLIST 100,200 Lists the statements between line numbers 100 and 200.

1 **MERGE**
2 **MERGE** "filename"
(effective for the manual operation in the PROgram or RUN mode)

Abbreviations: MER., MERG.

See also: CLOAD

Purpose

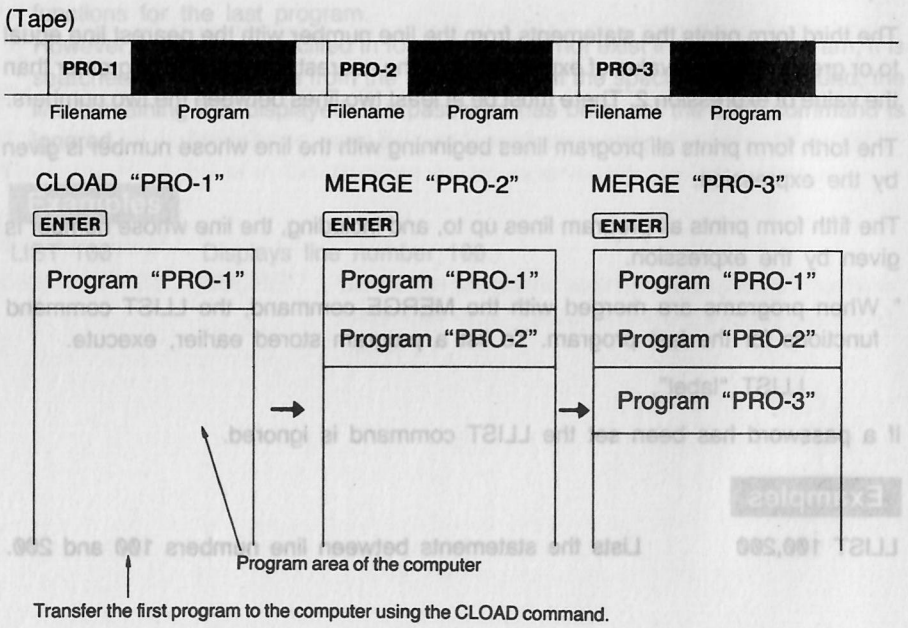
The MERGE command is used to load a program saved on cassette tape and merge it with the program existing in memory.

Use

The MERGE command retains the program already stored in the PC-1350 and then loads a program recorded on the tape. Therefore, several different programs can be stored in the PC-1350 at the same time.

Example

When programs with file names PRO-1, PRO-2 and PRO-3 are to be stored, PRO-1 is stored using the CLOAD command, and PRO-2 and PRO-3 are transferred to the computer using the MERGE command. The state of the storage is as follows.



Programs loaded using the MERGE command are stored as in the example. The programs are handled as follows by their line numbers.

- If the first line number of the program loaded using the MERGE command is larger than the last line number of the previously loaded program, the two programs are considered to be a single program.
- If the first line number of the program loaded using the MERGE command is smaller than the last line number of the previously loaded program, the two programs are considered separate.
  In the example above, where the line numbers for programs PRO-1, PRO-2, and PRO-3 are 10-200, 50-150, and 160-300 respectively, PRO-1 and PRO-2 are considered separate. PRO-2 and PRO-3 are considered to be a single program with line numbers 50-300.

* Loading programs with the MERGE command may result in 2 or more programs in the PC-1350 with the same line numbers. In this case the executed RUN or GOTO (RUN expression, GOTO expression) is valid only for the last merged program. There will be no way to execute the preceding program(s).
Therefore, put a label to the beginning of the program to be executed and execute using the defined key.
Note, however, that only the last merged program can be edited after the MERGE command has been executed and that the program(s) loaded earlier cannot be edited. Therefore, add the label to the program before merging the next program.

Merging password protected programs

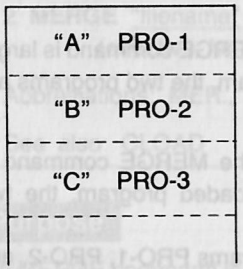
When loading programs with passwords (password protected programs) using the MERGE command, the handling of the programs differ as follows depending on whether the programs within the computer are protected.

When protected
Password protected programs cannot be loaded.

When not protected
If password protected programs are loaded using the MERGE command, all programs within the computer become protected.

When the programs within the computer are protected, even programs without passwords become password protected when loaded using the MERGE command.

Executing merged programs



The figure shows the memory when PRO-1 is loaded after which PRO-2 and PRO-3 are loaded using the MERGE command. If a program is started using RUN or GOTO (RUN expression or GOTO expression), PRO-3 will be executed. On the other hand, if started using RUN "label", GOTO "label", or a defined key, the specified label is searched for from the beginning of PRO-3 within the computer.

If not found in PRO-3, the search proceeds in PRO-1. If also not found in PRO-1, PRO-2 is searched. If the label is found, the program is executed from the labelled line.

Note that since the label is searched for in this manner, if a label used in PRO-1 and PRO-2 is also used in PRO-3, PRO-1 and PRO-2 cannot be executed.

Further, if the LIST command is executed with the label specified as LLIST "A", "C" [ENTER] , the label is searched for in PRO-3 first.

An error results since "C" appears first.

1 **NEW**

Abbreviations: none

Purpose

The NEW command is used to clear existing program or reserve memory.

Use

When used in the PROgram mode the NEW command clears all programs and data which are currently in memory. (The programs with password cannot be cleared.)

When used in the Reserve mode the NEW command clears all existing reserve memory.

The NEW command is not defined in the RUN mode and will result in an ERROR 9.

Examples

NEW Clears program or reserve memory

1 **PASS** "character string"

Abbreviations: PA., PAS.

See also: CSAVE, CLOAD, NEW

Purpose

The PASS command is used to set and cancel passwords.

Use

Passwords are used to protect programs from inspection or modification by other users. A password consists of a character string which is no more than seven characters long. The seven characters must be alphabetic or one of the following special symbols: ! # $ % & ( ) * + − / , . : ; < = > ? @ √ π ^

Once a PASS command has been given, the programs in memory are protected. A password protected program cannot be examined or modified in memory. It cannot be sent to tape or listed with LIST or LLIST, nor is it possible to add or delete program lines. If several programs are in memory and PASS is entered, all programs in memory are protected. The only way to remove this protection is to execute another PASS command with the same password.

Note: When a password with 7 or more characters is declared, only the first 7 characters are valid and are used to set and remove protection.
Press [ENTER] right after the password.
Writing characters or symbols after the password results in an error and the password cannot be cancelled.
(example) PASS"ABCDEFG":A=123 [ENTER] → Error

Examples

PASS "SECRET" Establishes the password 'SECRET' for all programs in memory.

* Regarding the Password When Using the RAM Card
The password for the PC-1350 and for the RAM card are separate.
When the RAM card is mounted, the password set, and the RAM card is removed, the program stored in the RAM card becomes protected. The password for the PC-1350 will not be set at this time.
Further, if the RAM card is mounted after protecting the program within the PC-1350 (i.e. setting the password), the program within the RAM card will not be protected.
(However, the password previously set for the program in the RAM card is valid.)

1 **RUN**
2 **RUN** line number

Abbreviatoins: R., RU.

See also: GOTO, MERGE

Purpose

The RUN command is used to execute a program in memory.

Use

The first form of the RUN command executes a program beginning with the lowest numbered statement in memory.

The second form of the RUN command executes a program beginning with the specified line number.

* When programs are merged with the MERGE command, the last merged program will be executed with format (1) or "RUN expression" in format (2).

RUN differs from GOTO in eight respects:

1) The value of the interval for WAIT is reset.
2) The display format established by USING statements is cleared.
3) Variables and arrays other than the fixed variables are cleared.
4) PRINT=PRINT status is set.
5) The pointer for READ is reset to the beginning DATA statement.
6) The cursor specification is cleared.
7) the setting for the graphic cursor is cleared to (0, 7).
8) Closes the serial I/O circuit (serial port).

Execution of a program with GOTO is identical to execution with the DEF key. In all three forms of program execution FOR/NEXT and GOSUB nesting is cleared.

Examples

RUN 100 Executes the program which begins at line number 100.

VERBS

1 **AREAD** variable name

Abbreviations: A., AR., ARE., AREA.

See also: INPUT verb and discussion of the use of the DEF key in Chapter 6

Purpose

The AREAD verb is used to read in a single value to a program which is started using the DEF key.

Use

When a program is labelled with a letter, so that it can be started using the DEF key, the AREAD verb can be used to enter a single starting value without the use of the INPUT verb. The AREAD verb must appear on the first line of the program following the label. If it appears elsewhere in the program, it will be ignored. Either a numeric or string variable may be used, but only one can be used per program.

To use the AREAD verb type the desired value in the RUN mode, press the DEF key, followed by the letter which identifies the program. If a string variable is being used, it is not necessary to enclose the entered string in quotes.

Examples

```
10 "X": AREAD N
20 PRINT N^2
30 END
```

Entering "7 DEF X" will produce a display of "49".

Notes:

1. When the display indicates PROMPT (">") at the start of program execution, the designated variable is cleared.

2. When the contents are displayed by PRINT verb at the start of program execution, the following is stored:

Example: When the program below is executed;

```
10 "A": PRINT "ABC", "DEFG"
20 "S": AREAD A$: PRINT A$
RUN mode
DEF [A] → ABC DEFG
DEF [S] → DEFG
```

- When the display indicated PRINT Numeric expression, Numeric expression, Numeric expression...., Numeric expression, or PRINT "String", "String", "String"..., the contents displayed last are stored.
- When the display indicated PRINT Numeric expression; numeric expression; Numeric expression..., the contents displayed first (on the extreme left) are stored.
- When the display indicates PRINT "String"; "String"; "String" ..., the contents of the "String" designated last (on the extreme right) are stored.

1 **BEEP** expression

Abbreviations: B., BE., BEE.

Purpose

The BEEP verb is used to produce an audible tone.

Use

The BEEP verb causes the PC-1350 to emit one or more audible tones at 4 kHz. The number of beeps is determined by the expression, which must be numeric. (Positive number less than 9.999999999E+99) The expression is evaluated, but only the integer part is used to determine the number of beeps.

BEEP may also be used as a command using numeric literals and predefined variables. In this case the beeps occur immediately after the [ENTER] key is pressed.

Examples

```
10 A=5 : B$="9"
20 BEEP 3          Produces 3 beeps.
30 BEEP A          Produces 5 beeps.
40 BEEP(A+4)/2     Produces 4 beeps.
50 BEEP B$         This is illegal and will produce an ERROR 9 message.
60 BEEP -4         Produces no beeps, but does not produce an error message.
```


1 **CHAIN**
2 **CHAIN** expression
3 **CHAIN** "filename"
4 **CHAIN** "filename", expression

Abbreviations: CH., CHA., CHAI.

See also: CLOAD, CSAVE, and RUN

Purpose

The CHAIN verb is used to start execution of a program which has been stored on cassette tape. It can only be used in connection with the optional CE-126P and CE-152.

Use

To use the CHAIN verb one or more programs must be stored on a cassette. Then, when the CHAIN verb is encountered in a running program, a program is loaded from the cassette and executed.

The first form of CHAIN loads the first program stored on the tape and begins execution with the lowest line number in the program. The effect is the same as having entered CLOAD and RUN when in the RUN mode.

The second form of CHAIN loads the first program stored on the tape and begins execution with the line number specified by the expression.

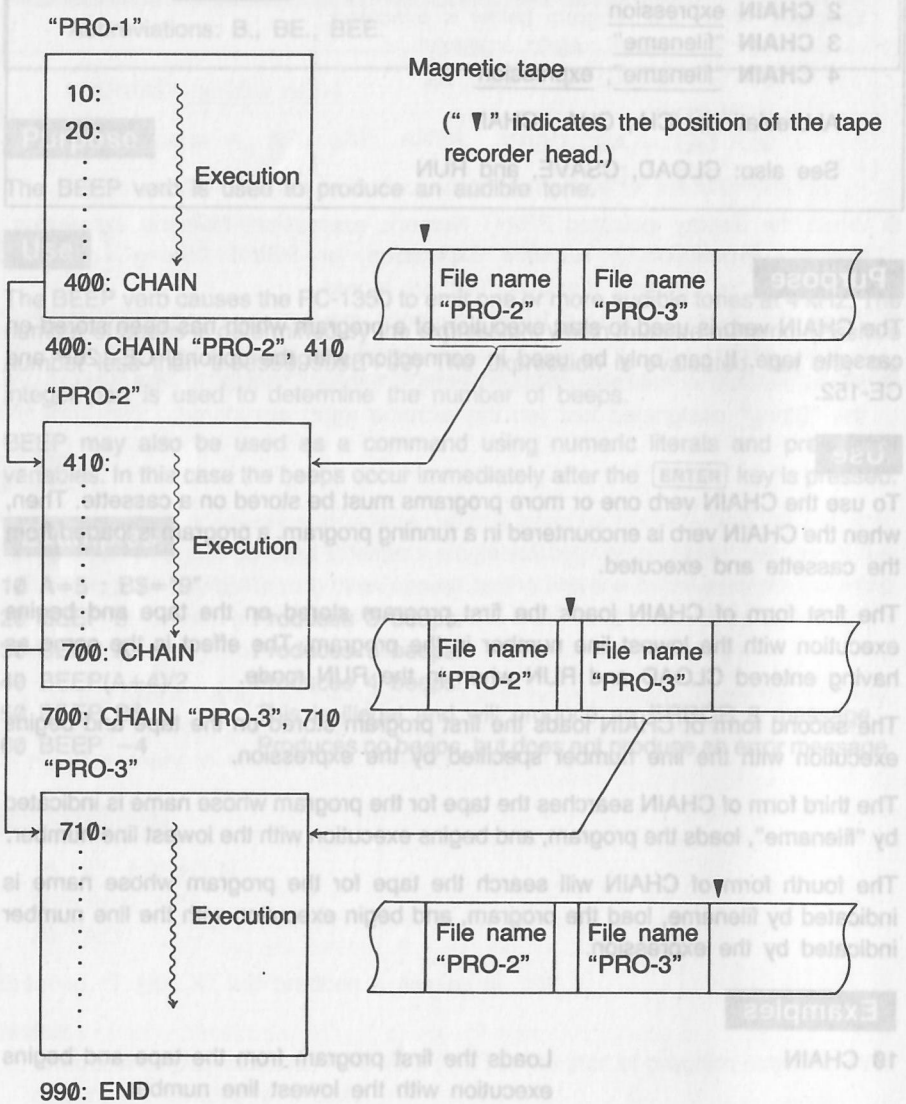
The third form of CHAIN searches the tape for the program whose name is indicated by "filename", loads the program, and begins execution with the lowest line number.

The fourth form of CHAIN will search the tape for the program whose name is indicated by filename, load the program, and begin execution with the line number indicated by the expression.

Examples

```
10 CHAIN              Loads the first program from the tape and begins
                      execution with the lowest line number.
20 CHAIN "PRO-2", 480 Searches the tape for a program named PRO-2,
                      loads it, and begins execution with line number 480.
```

For example, let's assume you have three program sections named PRO-1, PRO-2, PRO-3. Each of these sections ends with a CHAIN statement.



During execution, when the computer encounters the CHAIN statement, the next section is called into memory and executed. In this manner, all of the sections are eventually run.

Note: When a program containing a CHAIN command is loaded from the tape by the MERGE command, check to be sure that the CHAIN command is correct.

1 **CLEAR**

Abbreviations: CL., CLE., CLEA.

See also: DIM

Purpose

The CLEAR verb is used to erase all variables which have been used in the program and to reset all preallocated variables to zero or null.

Use

The CLEAR verb recovers space which is being used to store variables. This might be done when the variables used in the first part of a program are not required in the second part and available space is limited. CLEAR may also be used at the beginning of a program when several programs are resident in memory and you want to clear out the space used by execution of prior programs.

CLEAR does not free up the space used by the variables A–Z, A$–Z$, or A(1)–A(26) (without DIM declaration) since they are permanently assigned (see Chapter 4). CLEAR does reset numeric variables to zero and string variables to null.

Examples

```
10 A=5: DIM C(5)
20 CLEAR
```

Frees up the space assigned to C( ) and resets A to zero.

1 **CLS**

Abbreviations: none

See also: CURSOR

Purpose

The CLS command clears the display.

Use

Clears the display and returns the display start position to 0.

Examples

```
10:WAIT 3
20:INPUT A$
30:FOR B=0 TO 23
40:CLS
50:CURSOR B, 1
60:PRINT A$
70:NEXT B
80:CLS
90:END
```

This program displays the entry while moving it from left to right on the display unit (from the upper line to the lower line). Each time the FOR-NEXT loop of lines 30–70 is executed, the display is cleared with the CLS command, and display start position is shifted with the CURSOR command, and the contents of A$ are displayed with the PRINT command. By writing and clearing the display in this manner, the display can be made to appear to move. (Delete line 40 and execute. Note the difference.)

1 **CURSOR** expression 1, expression 2
2 **CURSOR** expression
3 **CURSOR**

Abbreviations: CU., CUR., CURS., CURSO.

See also: GCURSOR, CLS, INPUT, PRINT, PAUSE

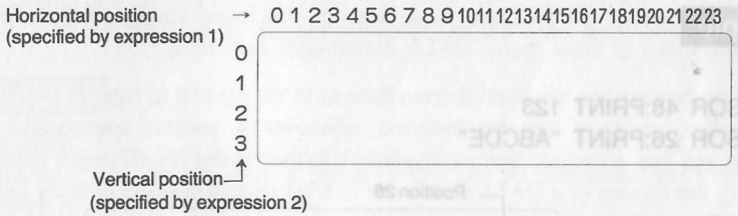
Purpose

Specifies the display start position in column units.

Use

Format (1) and format (2) specify the display start position in units of a character position for the contents displayed by the PRINT command, PAUSE command, etc.

* The display position is specified as follows using format (1).

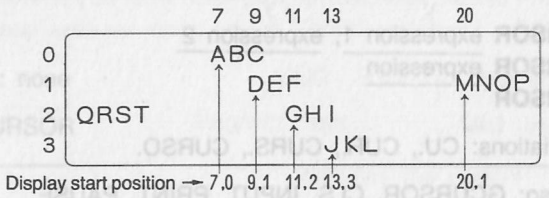

Horizontal position
(specified by expression 1)
→ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
0
1
2
3
Vertical position
(specified by expression 2)

A position on the display unit is specified by its horizontal and vertical positions. The values of expression 1 and expression 2 specify the horizontal and vertical position, respectively.
Therefore, be sure that the value of expression 1 is in the range of 0–23 and the value of expression 2 is in the range of 0–3.

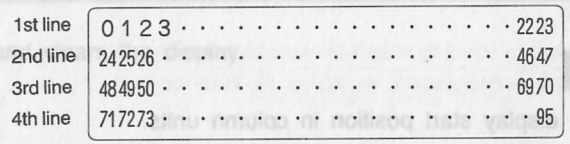
Example

```
 5 CLS
10 CURSOR 7,0:PRINT"ABC"
20 CURSOR 9,1:PRINT"DEF"
30 CURSOR 11,2:PRINT"GHI"
40 CURSOR 13,3:PRINT"JKL"
50 CURSOR 20,1:PRINT"MNOPQRST"
```

Executing the program will display the following.



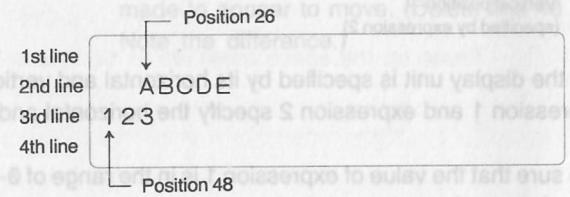
* The display position is specified as follows using format (2).



As shown, the positions on the display are assigned numbers from 0 to 95, starting from the top left of the display unit to the bottom right. The value of the expression in format (2) specifies the number of the position for the display start position. Therefore, be sure that the value of the expression in format (2) is within the range of 0–95. Exceeding this range results in a error (ERROR 9).

Example

```
 5 CLS
10 CURSOR 48:PRINT 123
20 CURSOR 26:PRINT "ABCDE"
```



* Format (3) clears the display start position.

The characters in front of and behind those displayed by the PRINT command or PAUSE command after the display start position was specified with the CURSOR command are retained. This feature can be used in many ways to change only a portion of the display. To clear the display, use the CLS command.

* The display start position specified in the CURSOR command can also be used for the INPUT command. However, after the INPUT command is executed, the display start position will still be specified at this position.
  Therefore, note that the characters will be displayed from the same position if a PRINT command is executed next.

Example

```
10 CLS
20 CURSOR 0,2
30 INPUT"DATA=";A
40 PRINT A/2
```

The position specified in line 20 applies for lines 30 and 40.

```
10 CLS
20 CURSOR 0,2
30 INPUT"DATA=";A
35 CURSOR
40 PRINT A/2
```

The position specified in line 20 is cleared.

* If the displayed characters overflow the screen, the screen is moved up (scrolled) to display all the characters even though the display start position was specified with the CURSOR command.

1 **DATA** expression list
Where: expression list is: expression
or: expression, expression list

Abbreviations: DA., DAT.

See also: READ, RESTORE

Purpose

The DATA verb is used to provide values for use by the READ verb.

Use

When assigning initial values to an array, it is convenient to list the values in a DATA statement and use a READ statement in a FOR ... NEXT loop to load the values into the array. When the first READ is executed, the first value in the first DATA statement is returned. Succeeding READs use succeeding values in the sequential order in which they appear in the program, regardless of how many values are listed in each DATA statement or how many DATA statements are used.

DATA statements have no effect if encountered in the course of regular execution of the program, so they can be inserted wherever it seems appropriate. Many programmers like to include them immediately following the READ which uses them. If desired, the values in a DATA statement can be read a second time by using the RESTORE statement.

Examples

```
10 DIM B(10)                Sets up an array.
20 WAIT 128
30 FOR I=1 TO 10
40 READ B(I)                Loads the values from the DATA statement into B( )
50 PRINT B(I)                 B(1) will be 10, B(2) will be 20, B(3) will be 30,
60 NEXT I                     etc.
70 DATA 10,20,30,40,50,60
80 DATA 70,80,90,100
90 END
```


1 **DEGREE**

Abbreviations: DE., DEG., DEGR., DEGRE.

See also: GRAD and RADIAN

Purpose

The DEGREE verb is used to change the form of angular values to decimal degrees.

Use

The PC-1350 has three forms for representing angular values—decimal degrees, radians and gradient. These forms are used in specifying the arguments to the SIN, COS, and TAN functions and in returning the results from the ASN, ACS, and ATN functions.

The DEGREE function changes the form for all angular values to decimal degree form until a GRAD or RADIAN verb is used. The DMS and DEG functions can be used to convert decimal degrees to degree, minute, second form and vice versa.

Examples

```
10 DEGREE
20 X=ASN 1        X now has a value of 90, i.e. 90 degrees, the Arcsine of 1.
30 PRINT X
```

1 **DIM** dim list

Where: dim list	is: dimension spec.
	or: dimension spec., dim list
and: dimension spec.	is: numeric dim spec.
	or: string dim spec.
and: numeric dim spec	is: numeric name (size)
and: string dim spec	is: string name (dims)
	or: string name (dims)*len
and: numeric name	is: valid numeric variable name
and: string name	is: valid string variable name
and: dims	is: size
	or: size, size
and: size	is: number of elements
and: len	is: length of each string in a string array

Abbreviations: D., DI.

Purpose

The DIM verb is used to reserve space for numeric and string array variables.

Use

Except for the array of the form; A( ), A$( ), two-character ( ), and two-character$ ( ), a DIM verb must be used to reserve space for any array variable.

The maximum number of dimensions in any array is two; the maximum size of any one dimension is 255. In addition to the number of elements specified in the dimension statement, one additional "zeroeth" element is reserved. For example, DIM B(3) reserves B(0), B(1), B(2), and B(3). In two dimensional arrays there is an extra "zeroeth" row and column.

In string arrays one specifies the size of each string element in addition to the number of elements. For example, DIM B$(3)*12 reserves space for 4 strings which are each a maximum of 12 characters long. If the length is not specified each string can contain a maximum of 16 characters.

When a numeric array is dimensioned, all values are initially set to zero; in a string array the values are set to null.

For the array A and A$ DIM declaration, refer to the paragraph discussing variables.

Array variables can be cleared (or set undefined) with the CLEAR command. When the program is started using the RUN command, array variables are automatically cleared.

The variable name once declared cannot be declared again. When a program once executed is executed again with the GOTO command on using the DEF key, the same variable name as formerly declared will be declared again if the line with the DIM command is executed. In this case, clear the array variable with the CLEAR command and the declare it again.

Examples

```
10 DIM B(10)
20 DIM C$(4, 4)*10
```

Reserves space for a numeric array with 11 elements.

Reserves space for a two dimensional string array with 5 rows and 5 columns; each string will be a maximum of 10 characters.

1 **END**

Abbreviations: E., EN.

Purpose

The END verb is used to signal the end of a program.

Use

When multiple programs are loaded into memory at the same time a mark must be included to indicate where each program ends so that execution does not continue from one program to another. This is done by including an END verb as the last statement in the program.

Examples

```
10 PRINT "HELLO"
20 END
30 PRINT "GOODBYE"
40 END
```

With these programs in memory a 'RUN 10' prints 'HELLO', but not 'GOODBYE'. 'RUN 30' prints 'GOODBYE'.

1 **FOR** numeric variable=expression 1 **TO** expression 2
2 **FOR** numeric variable=expression 1 **TO** expression 2
STEP expression 3

Abbreviations: F. and FO.; STE.

See also: NEXT

Purpose

The FOR verb is used in combination with the NEXT verb to repeat a series of operations a specified number of times.

Use

The FOR and the NEXT verbs are used in pairs to enclose a group of statements which are to be repeated. The first time this group of statements is executed the loop variable (the variable named immediately following the FOR) has the value of expression 1.

When execution reaches the NEXT verb the loop variable is increased by the step size and then this value is tested against expression 2. If the value of the loop variable is less than or equal to expression 2, the enclosed group of statements is executed again, starting with the statement following the FOR. In the first form the step size is 1; in the second form the step size is given by expression 3. If the value of the loop variable is greater than expression 2, execution continues with the statement which immediately follows the NEXT. Because the comparison is made at the end, the statements within a FOR/NEXT pair are always executed at least once.

Expression 1, expression 2, and expression 3 must be in the range of −9.999999999E99 ~ 9.999999999E99. When expression 3 is in zero FOR/NEXT loop will be infinite.

The loop variable may be used within the group of statements, for example as an index to an array, but care should be taken in changing the value of the loop variable.

Programs should be written so that they never jump from outside a FOR/NEXT pair to a statement within a FOR/NEXT pair. Similarly, programs must never leave a FOR/NEXT pair by jumping out. Always exit a FOR/NEXT loop via the NEXT statement. To do this, set the loop variable to a value higher than expression 2.

The group of statements enclosed by a FOR/NEXT pair can include another pair of FOR/NEXT statements which use a different loop variable as long as the enclosed pair is completely enclosed; i.e., if a FOR statement is included in the group, the matching NEXT must also be included. FOR/NEXT pairs may be "nested" up to five levels deep.

Examples

```
10 FOR I=1 TO 5
20 PRINT I
30 NEXT I
```

This group of statements prints the numbers 1, 2, 3, 4, 5.

```
40 FOR N=10 TO 0 STEP-1
50 PRINT N
60 NEXT N
```

This group of statements counts down 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0.

```
70 FOR N=1 TO 10
80 X=1
90 FOR F=1 TO N
100 X=X*F
110 NEXT F
120 PRINT X
130 NEXT N
```

This group of statements computes and prints N factorial for the numbers from 1 to 10.

Note: The execution of the FOR-NEXT loop does to end even if it jumps out of the loop. Therefore, note that a nesting error of the FOR-NEXT loop (ERROR 5) may result depending on the program (programs which execute the FOR command a number of times).

1 **GOSUB** expression

Abbreviations: GOS., GOSU.

See also: GOTO, ON ... GOSUB, ON ... GOTO, RETURN

Purpose

The GOSUB verb is used to execute a BASIC subroutine.

Use

When you wish to execute the same group of statements several times in the course of a program or use a previously written set of statements in several programs, it is convenient to use the BASIC capability for subroutines using the GOSUB and RETURN verbs.

The group of statements is included in the program at some location where they are not reached in the normal sequence of execution. A frequent location is following the END statement which marks the end of the main program. At those locations in the main body of the program—where subroutines are to be executed, include a GOSUB statement with an expression which indicates the starting line number of the subroutine. The last line of the subroutine must be a RETURN. When GOSUB is executed, the PC-1350 transfers control to the indicated line number and processes the statements until a RETURN is reached. Control is then transferred back to the statement following the GOSUB.

A subroutine may include a GOSUB. Subroutines may be "nested" in this fashion up to 10 levels deep.

The expression in a GOSUB statement may not include a comma, e.g., 'A(1, 2)' cannot be used. Since there is an ON ... GOSUB structure for choosing different subroutines at given locations in the program, the expression usually consists of just the desired line number. When a numeric expression is used it must evaluate to a valid line number, i.e., 1 to 65279, or an ERROR 4 will occur.

Example

```
10 GOSUB 100
20 END
100 PRINT "HELLO"
110 RETURN
```

When this program is run it prints the word 'HELLO' one time.

1 **GOTO** expression

Abbreviations: G., GO., GOT.

See also: GOSUB, ON ... GOSUB, ON ... GOTO

Purpose

The GOTO verb is used to transfer control to a specified line number.

Use

The GOTO verb transfers control from one location in a BASIC program to another location. Unlike the GOSUB verb, GOTO does not "remember" the location from which the transfer occurred.

The expression in a GOTO statement may not include a comma, e.g., 'A(1, 2)' cannot be used. Since there is an ON ... GOTO structure for choosing different destinations at given locations in the program, the expression usually consists of just the desired line number. When a numeric expression is used, it must evaluate to a valid line number, i.e., 1 to 65279, or an ERROR 4 will occur.

Well designed programs usually flow simply from beginning to end, except for subroutines executed during the program. Therefore, the principal use of the GOTO verb is as a part of an IF ... THEN statement.

Examples

```
10 INPUT A$
20 IF A$="Y" THEN GOTO 50
30 PRINT "NO"
40 GOTO 60
50 PRINT "YES"
60 END
```

This program prints 'YES' if a 'Y' is entered and prints 'NO' if anything else is entered.

1 **GRAD**

Abbreviations: GR., GRA.

See also: DEGREE and RADIAN

Purpose

The GRAD verb is used to change the form of angular values to gradient form.

Use

The PC-1350 has three forms for representing angular values—decimal degrees, radians, and gradient. These forms are used in specifying the arguments to the SIN, COS, and TAN functions and in returning the results from the ASN, ACS, and ATN functions.

The GRAD function changes the form for all angular values to gradient form until a DEGREE or RADIAN verb is used. Gradient form represents angular measurement in terms of percent gradient, i.e., a 45° angle is a 50^g gradient.

Examples

```
10 GRAD
20 X=ASN 1
30 PRINT X
```

X now has a value of 100, i.e., a 100^g gradient, the Arcsine of 1.

1 **IF** condition **THEN** statement
2 **IF** condition statement

Abbreviations: none for IF, T., TH., THE.

Purpose

The IF ... THEN verb pair is used to execute or not to execute a statement depending on conditions at the time the program is run.

Use

In the normal running of BASIC programs, statements are executed in the sequence in which they occur. The IF ... THEN verb pair allows decisions to be made during execution so that a given statement is executed only when desired. When the condition part of the IF statement is true, the statement is executed; when it is false, the statement is skipped.

The condition part of the IF statement can be any relational expression as described in Chapter 4. It is also possible to use a numeric expression as a condition, although the intent of the statement will be less clear. Any expression which evaluates to zero or a negative number is considered false; any which evaluates to a positive number is considered true.

The statement which follows the THEN may be any BASIC statement, including another IF ... THEN. If it is a LET statement, the LET verb itself must appear. Unless the statement is an END, GOTO, or ON ... GOTO, the statement following the IF ... THEN statement is the next one executed regardless of whether the condition is true.

The two forms of the IF statement are identical in action, but the first form is clear.

Examples

```
10 INPUT "CONTINUE?"; A$
20 IF A$="YES" THEN GOTO 10
30 IF A$="NO" THEN GOTO 60
40 PRINT "YES OR NO, PLEASE"
50 GOTO 10
60 END
```

This program continues to ask 'CONTINUE?' as long as 'YES' is entered; it stops if 'NO' is entered, and complains otherwise.

1 **INPUT** input list

Where: input list	is:	input group
	or:	input group, input list
and: input group	is:	var list
	or:	prompt, var list
	or:	prompt; var list
and: var list	is:	variable
	or:	variable, var list
and: prompt	is:	any string constant

Abbreviations: I., IN., INP., INPU.

See also: INPUT #, READ, CURSOR, PRINT

Purpose

The INPUT verb is used to enter one or more values from the keyboard.

Use

When you want to enter different values each time a program is run, use the INPUT verb to enter these values from the keyboard.

In its simplest form the INPUT statement does not include a prompt string; instead a question mark is displayed on the left edge of the display. A value is then entered, followed by the ENTER key. This value is assigned to the first variable in the list. If other variables are included in the same INPUT statement, this process is repeated until the list is exhausted.

If a prompt is included in the INPUT statement, the process is exactly the same except that, instead of the question mark, the prompt string is displayed at the left edge of the display. If the prompt string is followed by a semicolon, the cursor is positioned immediately following the prompt. If the prompt is followed by a comma, the prompt is displayed. Then when a key is pressed, the display is cleared and the first character of the input is displayed at the left edge.

When a prompt is specified and there is more than one variable in the list following it, the second and succeeding variables are prompted with the question mark. If a second prompt is included in the list, it is displayed for the variable which immediately follows it.

When the display starting position has been specified using the CURSOR command

before executing the INPUT command, the input prompt or "?" will be displayed from that position.

If the [ENTER] key is pressed and no input is provided, the variable retains the value it had before the INPUT statement.

Examples

10 INPUT A	Clears the display and puts a question mark at the left edge.
20 INPUT "A=";A	Displays 'A=' and waits for input data.
30 INPUT "A=",A	Displays'A='. When data is input 'A=' disappears and the data is displayed starting at left edge.
40 INPUT "X=?";X,"Y=?";Y	Displays 'X=?' and waits for first input. After [ENTER] is pressed, display is cleared and 'Y=?' is displayed at left edge.

Notes:

Clear the error during input for the INPUT command by pressing the [CLS] key and then input the correct data. When playing back using the [◄] or [►] key, unnecessary displays may appear.

The data input proceeds normally even if this occurs.

1 **INPUT #** var list
2 **INPUT #** "filename"; var list

Where: var list is: variable
or: variable, var list

Abbreviations: I. #, IN. #, INP. #, INPU. #

See also: INPUT, PRINT #, READ

Purpose

The INPUT # verb is used to enter values from the cassette tape.

Use and Examples

The following variable types can be specified in the INPUT # statement:

(1) Fixed variables—A, B, C, A(7), D*, A(20)*, etc.
(2) Simple variables—AA, B3, CP$, etc.
(3) Array variables—S(*), HP(*), K$(*), etc.

1) Transferring data to fixed variables

To transfer data from tape to fixed variables, specify the variable names in the INPUT # statement.

```
INPUT # "DATA 1" ; A, B, X, Y
```

This statement transfers data from the cassette file named "DATA 1" to the variables A, B, X, and Y in that order.

To fill all the available fixed variables and, if defined, extended variables (A(27) and beyond) with data transferred from tape, specify the first variable with an asterisk (*) subscripted to it.

```
INPUT # "D-2"; D*
```

This statement transfers the contents of the tape file "D-2" to variables D through Z and to A(27) and beyond.

```
INPUT # A(10)* (without DIM declaration)
```

This statement transfers the data of the first file found after the tape was started, to the variables A(10) and beyond (to J through Z and A(27) and beyond).

Note 1. If an array named A is already defined by the DIM statement, it is not possible to define subscripted fixed variables in the form of A( ).

Note 2. Data transfer to fixed variables and extended variables (A(27) and beyond) will continue until the end of the source data file on the tape is reached, but if the computer's memory becomes full, an error (ERROR 6) results.

2) Data transfer to simple variables

Data in a tape file can be transferred to simple variables by specifying the desired simple variable names in the INPUT # statement.

```
INPUT # "DM-1"; AB, Y1, XY$
```

This statement transfers data from the tape file named "DM-1" to simple variables AB, Y1, and XY$.

Note 1. Numeric data must be transferred to numeric simple variables, and character data must be simple character variables. Cross-transfer is not allowed.

Note 2. Locations for simple variables must be set aside in the program data area before the INPUT # statement is executed. If not, an error will result. Use assignment statements to reserve the locations for simple variables.

```
AA=0 [ENTER]
B1$="A" [ENTER]

INPUT # AA, B1$ [ENTER]
```

Use appropriate numeric values or characters in assignment statements to reserve locations for variables.

3) Data transfer to array variables

To transfer data from a tape file to array variables, specify the array name in the INPUT # statement in the form of array name(*).

```
50 DIM B(5)
60 INPUT # "DS-4"; B(*)
```

This statement transfers data from the tape file named "DS-4" to the variables (B(0) through B(5)) in array B.

Note 1. Numeric data must be transferred to numeric array variables with the same length as that of the data, character data must be transferred to character array variables with the same length as that of the data. If this rule is not observed, an error will result.

Note 2. Locations for array variables must be set aside in the program data area before the INPUT # statement is executed. If not, an error will result. Use the DIM statement to define the array in advance.

–CAUTION–

If the number of variables specified in the INPUT # statement does not agree with the amount of data recorded on the tape, the following will happen:

* If the number of pieces of data recorded on the tape file (to be transferred) is greater than the number of specified variables, data transfer will be performed to the last variable, and the remaining data will be ignored.
* If the number of pieces of data recorded in the tape file (to be transferred) is smaller than the number of specified variables, all the file data will be transferred to the variables to the end of the file, and the remaining variables will maintain their previous contents. In this case, however, the computer will continue to wait for data transfer from the tape. To halt this state, you should operate the [ON BRK] key.
* If the INPUT # statement is executed with no variable name specified in it, an error (ERROR 1) will result.

1 **LET** variable=expression
2 variable=expression

Abbreviations: LE.

Purpose

The LET verb is used to assign a value to a variable.

Use

The LET verb assigns the value of the expression to the designated variable. The type of the expression must match that of the variable, i.e. only numeric expressions can be assigned to numeric variables and only string expressions can be assigned to string variables. In order to convert from one type to the other, one of the explicit type conversion functions, STR$ or VAL, must be used.

The LET verb may be omitted in all LET statements except those which appear in the THEN clause of an IF ... THEN statement. In this one case the LET verb must be used.

Examples

```
10 I=10                          Assignes the value 10 to I.
20 A=5*I                         Assigns the value 50 to A.
30 X$=STR$ (A)                   Assigns the value '50' to X$
40 IF I>=10 THEN LET Y$=X$+".00" Assigns the value '50.00' to Y$.
```


1 **LPRINT** print expr
2 **LPRINT** print expr, print expr, ..., print expr
3 **LPRINT** print list
4 **LPRINT** print list;
5 **LPRINT**

Where: print list is: print expr
or: print expr; print list
and: print expr is: expression
or: USING clause; expression

The USING clause is described separately under USING

Abbreviations: LP., LPR., LPRI., LPRIN.

See also: PAUSE, PRINT, USING, and WAIT

Purpose

The LPRINT verb is used to print information on the printer of the optional CE-126P.

Use

When the serial I/O interface is open due to the OPEN command, the LPRINT command outputs the program at the serial I/O interface terminal. (See page 185.) To return the program printing command to the printer (CE-126P), execute the CLOSE command.

The LPRINT verb is used to print prompting information, results of calculations, etc. The first form of the LPRINT statement prints a single value. If the expression is numeric, the value will be printed at the far right edge of the paper. If it is a string expression, the print is made starting at the far left.
In format (2), the number of print columns is delimited into groups of 12 columns. The specified values are printed in sequence. In other words, the first specified value is printed on the left side of the first line, the second specified value on the right side of the first line, the third specified value on the left side of the second line, and the fourth specified value on the right side of the second line.

The numeric value within the 12 column (digit) range is printed at the far right end of the display, while the character value (string value) is printed starting at the far left. If the value to be printed exceeds 12 columns, numeric values are printed with the least significant digit(s) of the decimal fraction truncated so value is within 12 digits, and characters are printed from the first 12 (from the left).

Note 1: The number of the values (items) specified in format (2) must be within 2–8.

Note 2: Even if the display starting position has been specified in format (2) with format (4) or the CURSOR command, the specification will be cleared and printing will be performed in the form as shown above. The values are printed from the left edge of the paper.

In format (3), the values are printed from the left edge of the paper. If the value to be printed exceeds 24 columns, a newline is automatically performed. Up to a maximum of 96 characters can be printed. An error occurs if the 96th column is in the middle of a numeric value.

In format (4), at the specified printing value, the value specified in the LPRINT command to be executed next will be printed in succession. However, due to the structure of the printer, printing occurs in the following instances:

(1) when the value to be printed exceeds 24 columns.
(2) when an LPRINT command not ending with a ";" has been executed.
(3) when the program execution ends.

In format (5), no printing occurs but the paper is fed one line.

Examples

```
10 A=10:B=20:X$="ABCDE":Y$="XYZ"
20 LPRINT A
30 LPRINT X$
40 LPRINT A,B,X$,Y$
50 LPRINT X$;A;B
60 LPRINT
70 LPRINT A*B;
80 LPRINT Y$
```


1 **NEXT** numeric variable

Abbreviations: N., NE., NEX.

See also: FOR

Purpose

The NEXT verb is used to mark the end of a group of statements which are being repeated in a FOR/NEXT loop.

Use

The use of the NEXT verb is described under FOR. The numeric variable in a NEXT statement must match the numeric variable in the corresponding FOR.

Examples

```
10 FOR I=1 TO 10
20 PRINT I
30 NEXT I
```

Print the numbers from 1 to 10 each time the ENTER is pressed.

1 **ON** expression **GOSUB** expression list

Where: expression list is: expression
or: expression, expression list

Abbreviations: O.; GOS., GOSU.

See also: GOSUB, GOTO, ON ... GOTO

Purpose

The ON ... GOSUB verb is used to execute one of a set of subroutines depending on the value of a control expression.

Use

When the ON ... GOSUB verb is executed the expression between ON and GOSUB is evaluated and reduced to an integer. If the value of the integer is 1, the first subroutine in the list is executed as in a normal GOSUB. If the expression is 2, the second subroutine in the list is executed, and so forth. After the RETURN from the subroutine execution proceeds with the statement which follows the ON ... GOSUB.

If the expression is zero, negative, or larger than the number of subroutines provided in the list, no subroutine is executed and execution proceeds with the next line of the program.

NOTE: Commas may not be used in the expressions following the GOSUB. The PC-1350 cannot distinguish between commas in expressions and commas between expressions.

Examples

```
10 INPUT A
20 ON A GOSUB 100, 200, 300
30 END
100 PRINT "FIRST"
110 RETURN
200 PRINT "SECOND"
210 RETURN
300 PRINT "THIRD"
310 RETURN
```

An input of 1 prints "FIRST"; 2 prints "SECOND"; 3 prints "THIRD". Any other input does not produce any print.

1 **ON** expression **GOTO** expression list

Where: expression list is: expression
or: expression, expression list

Abbreviations: O.; G., GO., GOT.

See also: GOSUB, GOTO, ON ... GOSUB

Purpose

The ON ... GOTO verb is used to transfer control to one of a set of locations depending on the value of a control expression.

Use

When the ON ... GOTO verb is executed the expression between ON and GOTO is evaluated and reduced to an integer. If the value of the integer is 1, control is transferred to the first location in the list. If the expression is 2, control is transferred to the second location in the list; and so forth.

If the expression is zero, negative, or larger than the number of locations provided in the list, execution proceeds with the next line of the program.

NOTE: Commas may not be used in the expressions following the GOTO. The PC-1350 cannot distinguish between commas in expressions and commas between expressions.

Examples

```
10 INPUT A
20 ON A GOTO 100,200,300
30 GOTO 900
100 PRINT "FIRST"
110 GOTO 900
200 PRINT "SECOND"
210 GOTO 900
300 PRINT "THIRD"
310 GOTO 900
900 END
```

An input of 1 prints 'FIRST'; 2 prints 'SECOND'; 3 prints 'THIRD'. Any other input does not produce any print.

1 **PAUSE** print expr
2 **PAUSE** print expr, print expr, ..., print expr
3 **PAUSE** print list
4 **PAUSE** print list;
5 **PAUSE**

Where: print list is: print expr
or: print expr; print list
and: print expr is: expression
or: USING clause; expression

The USING clause is described separately under USING

Abbreviations: PAU., PAUS.

See also: LPRINT, PRINT, CURSOR, USING, and WAIT

Purpose

The PAUSE verb is used to print information on the display for a short period.

Use

The PAUSE verb is used to display prompting information, results of calculations, etc. The operation of PAUSE is identical to PRINT except that after PAUSE the PC-1350 waits for short preset interval of about .85 seconds and then continues execution of the program without waiting for the ENTER key or the WAIT interval.

The first form of the PAUSE statement displays a single value. If the expression is numeric, the value is printed at the far right end of the display. If it is a string expression, the display is made starting at the far left.

However, when the display starting position is specified using format (4) or the CURSOR command, the display starts from that position.
In format (2), the display unit is divided into groups of 12 columns. The values are displayed, in sequence, from the first specified value.

In this case too, within a range of 12 columns, the numeric value of an expression is displayed from the right end of the display and characters are displayed from the left side.

- The number of the values (items) specified in format (2) must be within 2—8.
- If the specified value exceeds 12 columns, the following is performed.
  1) When the numeric value exceeds 12 digits (when the decimal fraction in the exponential display is 8 digits or more), the least significant digit(s) is truncated.
  2) When the characters exceed 12 columns, only the first 12 characters (from the left) are displayed.

In format (3), the specified value is displayed continuously from the left side of the display. However, if the display starting position has been specified using format (4) or the CURSOR command, the display starts from that position.

Note: If the value to be displayed in format (3) exceeds 96 columns, the excess portion is not displayed. When the value to be displayed does exceed 96 columns, an error (ERROR 6) occurs if the 96 column is in the middle of a numeric value.

In format (4), the specified value is displayed from the left side of the display. The column following the end of this displayed value is specified as the display starting position for display commands such as for the next PRINT command.
In format (5), the previously displayed value is displayed as is.

Examples

```
10 A=10:B=20:X$="ABCDEF":
   Y$="XYZ"
```

	Display
20 PAUSE A	10.
30 PAUSE X$	ABCDEF
40 PAUSE X$, Y$, A, B	ABCDEF XYZ 10. 20.
50 PAUSE Y$;X$;	XYZABCDEF
60 PAUSE A*B	XYZABCDEF200.

1 **PRINT** print expr
2 **PRINT** print expr, print expr, print expr, print expr
3 **PRINT** print list
4 **PRINT** print list;
5 **PRINT**
6 **PRINT=LPRINT**
7 **PRINT=PRINT**

Where: print list is: print expr
or: print expr; print list
and: print expr is: expression
or: USING clause; expression

The USING clause is described separately under USING

Abbreviations: P., PR., PRI., PRIN.

See also: LPRINT, PAUSE, CURSOR, USING, and WAIT

Purpose

The PRINT verb is used to print information on the display or on the printer of the CE-126P.

Use

The PRINT verb is used to display prompting information, results of calculations, etc. The first form of the PRINT statement displays a single value. If the expression is numeric, the value is printed at the far right end of the display. If it is a string expression, the display is made starting at the far left.

However, when the display starting position is specified using format (4) or the CURSOR command, the display starts from that position. In format (2), the display unit is divided into groups of 12 columns. The values are displayed, in sequence, from the first specified value. In this case too, within a range of 12 columns, the value of an expression is displayed from the right end of the display and characters are displayed from the left side.

- The number of the values (items) specified in format (2) must be within 2–8.
- If the specified value exceeds 12 columns, the following is performed.
  1) When the numeric value exceeds 12 digits (when the decimal fraction in the exponential display is 8 or more), the least significant digit(s) is truncated.
  2) When the characters exceed 12 columns, only the first 12 characters (from the left) are displayed.

In format (3), the specified value is displayed continuously from the left side of the display. However, if the display starting position has been specified using format (4) or the CURSOR command, the display starts from that position.

Note: If the value to be displayed in format (3) exceeds 96 columns, the excess portion is not displayed. When the value to be displayed does exceed 96 columns, an error (ERROR 6) occurs if the 96 column is in the middle of a numeric value.

In format (4), the specified value is displayed from the left side of the display. The column following the end of this displayed value is specified as the display starting position of display commands such as for the next PRINT command.

Note: Do not combine display commands (PRINT, etc.) with serial I/O commands (LPRINT, etc. for the serial I/O interface).
Combining them may clear the display start position specified in format (4).

In format (5), the previously displayed value is displayed as is.

The sixth and seventh forms of the PRINT statement do no printing. The sixth form causes all PRINT statements which follow it in the program to be treated as if they were LPRINT statements. The seventh form resets this condition so that the PRINT statements will again work with the display.

Examples

```
10 A=123:B=5/9:X$="ABCDEF":
   Y$="VWXYZ"
20 PRINT X$,B
```

Display

```
ABCDEF          5.55555E-01
```

```
30 PRINT A;B
```

```
123.5.555555556E-01
```

```
40 PRINT X$;A;
```

```
ABCDEF123.
```

```
50 PRINT Y$;B
```

```
ABCDEF123.VWXYZ5.5555555
56E-01
```

1 **PRINT #** " var list"
2 **PRINT #** "filename" ; var list

Where: var list is: variable
or: variable, var list

Abbreviations: P. #, PR. #, PRI. #,PRIN. #

See also: INPUT #, PRINT, READ

Purpose

The PRINT # verb is used to store values on the cassette tape.

Use and Examples

The following variable types can be used for variable names:
(1) Fixed variables—A, B, X, A(26), C*, A(10)*,etc.
(2) Simple variables—AA, B2, XY$, etc.
(3) Array variables—B(*), CD(*), N$(*), etc.

1) Saving fixed variable contents onto tape

The contents of fixed variables can be saved onto tape by specifying the desired variable names (separated by commas) in the PRINT # statement.

PRINT # "DATA 1"; A, B, X, Y

This statement saves contents of variables A, B, X, and Y into tape file named "DATA 1".

If you wish to save the contents of the specified fixed variable and all the subsequent fixed variables, subscript that variable name with an asterisk*.

PRINT # "D-2"; D* This statement saves the contents of fixed variables D through Z (and of extended variables A(27) and beyond, if defined) into the tape file named "D-2".

PRINT # E, X$, A(30)* This statement saves the contents of the fixed variables E and X$ and of the extended variables A(30) and all the remaining variables, onto the tape without filename.

Note: Subscripted fixed variable names A(1) through A(26) can be specified in the PRINT # statement in much the same way as A through Z (or A$ through Z$). However, if array A is already defined by the DIM statement, A( ) cannot be used to define subscripted fixed variables.

2) Saving simple variable (two-character variable) contents

The contents of simple variables can be saved onto tape by specifying the desired variable names.

PRINT # "DM-1"; AB, Y1, XY$

This statement saves the contents of the simple variables AB, Y1, and XY$ into the tape file named 'DM-1'.

3) Saving array variable contents

The contents of all variables of a specific array can be saved onto tape by specifying the array name subscripted by an asterisk enclosed in parentheses(*).

PRINT # "DS-2";X(*),Y$(*)

This statement saves the contents of all the elements (X(0), X(1), ...) of the array X, and of all the elements (Y$(0), Y$(1), ...) of the array Y$, into the tape file name 'DS-2'.

Note: It is not possible to save the contents of only one or more specific elements of an array. While fixed variables or subscripted fixed variables allow you to save only specific parts of them, an array (such as A), defined by the DIM statement does not allow you to save only a specific part of it.

* If the PRINT # statement is executed with no variable names specified, an error (ERROR1) will result.

—CAUTION—

The locations for extended variables such as A(27) and beyond, simple variables, and/or array variables must be set aside in the program/data area before the PRINT # statement is executed. Otherwise, the execution of the PRINT # statement for undefined variables will result in an error.

1 **RADIAN**

Abbreviations: RAD., RADI., RADIA.

See also: DEGREE and GRAD

Purpose

The RADIAN verb is used to change the form of angular values to radian form.

Use

The PC-1350 has three forms for representing angular values—decimal degrees, radians, and gradient. These forms are used in specifying the arguments to the SIN, COS, and TAN functions and in returning the results from the ASN, ACS, and ATN functions.

The RADIAN function changes the form for all angular values to radian form until a DEGREE or GRAD verb is used. Radian form represents angles in terms of the length of the arc with respect to a radius, i.e., 360° is 2 PI radians since the circumference of a circle is 2 PI times the radius.

Examples

```
10 RADIAN
20 X=ASN 1
30 PRINT X
```

X now has a value of 1.570796327 or PI/2, the Arcsine of 1.

1 **RANDOM**

Abbreviations: RA., RAN., RAND., RANDO.

Purpose

The RANDOM verb is used to reset the seed for random number generation.

Use

When random numbers are generated using the RND function, the PC-1350 begins with a predetermined "seed" or starting number. The RANDOM verb resets this seed to a new randomly determined value.

The starting seed will be the same each time the PC-1350 is turned on, so the sequence of random numbers generated with RND is the same each time, unless the seed is changed. This is very convenient during the development of a program because it means that the behavior of the program should be the same each time it is run even though it includes a RND function. When you want to have the numbers be truly random, the RANDOM statement can be used to make the seed itself random.

Examples

```
10 RANDOM
20 X=RND 10
```

When run from line 20, the value of X is based on the standard seed. When run from line 10, a new seed is used.

1 **READ** variable list
Where: variable list is: variable
or: variable , variable list

Abbreviations: REA.

See also: DATA, RESTORE

Purpose

The READ verb is used to read values from a DATA statement and assign them to variables.

Use

When assigning initial values to an array, it is convenient to list the values in a DATA statement and use a READ statement in a FOR ... NEXT loop to load the values into the array. When the first READ is executed, the first value in the first DATA statement is returned. Succeeding READs use succeeding values in the sequential order in which they appear in the program, regardless of how many values are listed in each DATA statement or how many DATA statements are used.

If desired, the values in a DATA statement can be read a second time by using the RESTORE statement.

Examples

```
10 DIM B (10)              Set up an array
20 WAIT 32
30 FOR I=1 TO 10
40 READ B(I)               Loads the values from the DATA statement into
50 PRINT B(I)*2;             B( )—B(1) is 10, B(2) is 20, B(3) is 30, etc.
60 NEXT I
70 DATA 10, 20, 30, 40, 50, 60
80 DATA 70, 80, 90, 100
90 END
```


1 **REM** remark

Abbreviations: none

Purpose

The REM verb is used to include comments in a program.

Use

Often it is useful to include explanatory comments in a program. These can provide titles, names of authors, dates of last modification, usage notes, reminders about algorithms used, etc. These comments are included by means of the REM statement.

The REM statement has no effect on the program execution and can be included anywhere in the program. Everything following the REM verb in that line is treated as a comment.

Examples

```
10 REM THIS LINE HAS NO EFFECT
```

1 **RESTORE**
2 **RESTORE** expression

Abbreviations: RES., REST., RESTO., RESTOR.

See also: DATA, READ

Purpose

The RESTORE verb is used to reread values in a DATA statement or to change the order in which these values are read.

Use

In the regular use of the READ verb the PC-1350 begins reading with the first value in a DATA statement and proceeds sequentially through the remaining values. The first form of the RESTORE statement resets the pointer to the first value of the first DATA statement, so that it can be read again. The second form of the RESTORE statement resets the pointer to the first value of the first DATA statement whose line number is greater than the value of the expression.

Examples

```
10 DIM B(10)
20 WAIT 32
30 FOR I=1 TO 10
40 RESTORE
50 READ B(I)
60 PRINT B(I)*I;
70 NEXT I
80 DATA 20
90 END
```

Sets up an array.

Assign the value 20 to each of the elements of B ( ).

1 **RETURN**

Abbreviations: RE., RET., RETU., RETUR.

See also: GOSUB, ON ... GOSUB

Purpose

The RETURN verb is used at the end of a subroutine to return control to the statement following the originating GOSUB.

Use

A subroutine may have more than one RETURN statement, but the first one executed terminates the execution of the subroutine. The next statement executed will be the one following the GOSUB or ON ... GOSUB which calls the subroutine. If a RETURN is executed without a GOSUB, an ERROR 5 will occur.

Examples

```
10 GOSUB 100
20 END
100 PRINT "HELLO"
110 RETURN
```

When run this program prints the word "HELLO" one time.

1 **STOP**

Abbreviations: S., ST., STO.,

See also: END; CONT command

Purpose

The STOP verb is used to halt execution of a program for diagnostic purposes.

Use

When the STOP verb is encountered in program execution, the PC-1350 execution halts and a message is displayed such as 'BREAK IN 200' where 200 is the number of the line containing the STOP. STOP is used during the development of a program to check the flow of the program or examine the state of variables. Execution may be restarted using the CONT command.

Examples

```
10 STOP
```

Causes "BREAK IN 10" to appear in the display.

1 **TROFF**

Abbreviations: TROF.

See also: TRON

Purpose

The TROFF verb is used to cancel the trace mode.

Use

Execution of the TROFF verb restores normal execution of the program.

Examples

```
10 TRON
20 FOR I=1 TO 3
30 NEXT I
40 TROFF
```

When run, this program displays the line numbers 10, 20, 30, 30, 30 and 40 as the [↓] is pressed.

1 **TRON**

Abbreviations: TR., TRO.

See also: TROFF

Purpose

The TRON verb is used to initiate the trace mode.

Use

The trace mode provides assistance in debugging programs. When the trace mode is on, line number of each statement is displayed after each statement is executed. The PC-1350 then halts and waits for the Down Arrow key to be pressed before moving on to the next statement. The Up Arrow key may be pressed to see the statement which has just been executed. The trace mode continues until a TROFF verb is executed or the key operation of the [SHIFT] and [CA / CLS], is performed.
After a result is displayed at the position specified in the CURSOR command during the trace mode, the next line number is displayed on the next line of the display. (See page 113 on the CURSOR command.)
When, during the trace mode, the display start position is specified after execution of the CURSOR command, it is cleared if variables are called or calculations are made in manual operation.

Examples

```
10 TRON
20 FOR I=1 TO 3
30 NEXT I
40 TROFF
```

When run, this program displays the line numbers 10, 20, 30, 30, 30 and 40 as the [↓] is pressed.

1 **USING**
2 **USING** "editing specification"
3 **USING** character variable

Abbreviations: U., US., USI., USIN.

See also: LPRINT, PAUSE, PRINT

Further guide to the use of USING is provided in Appendix C

Purpose

The USING verb is used to control the format of displayed or printed output.

Use

The USING verb can be used by itself or as a clause within a LPRINT, PAUSE, or PRINT statement. The USING verb establishes a specified format for output which is used for all output which follows until changed by another USING verb.

The editing specification of the USING verb consists of a quoted string composed of some combination of the following editing characters:

- # Right justified numerfc field character
- . Decimal point.
- ^ Used to indicate that numbers should be displayed in scientific notation.
- & Left justfied alphanumeric field.

For example, "####" is an editing specification for a right justified numeric field with room for 3 digits and the sign. In numeric fields, a location must be included for the sign, even if it will always be positive.

Editing specifications may include more than one field. For example "####&&&&" could be used to print a numeric and a character field next to each other.

If the editing specification is missing, as in format 1, special formatting is turned off and the built-in display rules pertain.

Examples

	Display
10 A=125: X$="ABCDEF"	
20 PRINT USING "##.##^";A	1.25E 02
30 PRINT USING "&&&&&&&&";X$	ABCDEF
40 PRINT USING "####&&&";A;X$	125ABC

1 **WAIT** expression
2 **WAIT**
Abbreviations: W., WA., WAI.
See also: PAUSE, PRINT

Purpose

The WAIT verb is used to control the length of time that displayed information is shown before program execution continues.

Use

In normal execution the PC-1350 halts execution after a PRINT, GPRINT, PSET, PRESET, LINE command until the [ENTER] key is pressed. The WAIT command causes the PC-1350 to display for a specified interval and then proceed automatically (similar to the PAUSE verb). The expression which follows the WAIT verb determines the length of the interval. The interval may be set to any value from 0 to 65535. Each increment is about one fifty-nine of a second. WAIT 0 is too fast to be read reasonably; WAIT 65535 is about 19 minutes. WAIT with no following expression resets the PC-1350 to the original condition of waiting until the [ENTER] key is pressed.

Examples

10 WAIT 59 Causes PRINT to wait about 1 second.

FUNCTIONS

Pseudovariables

Pseudovariables are a group of functions which take no argument and are used like simple variables wherever required.

1 **INKEY$**

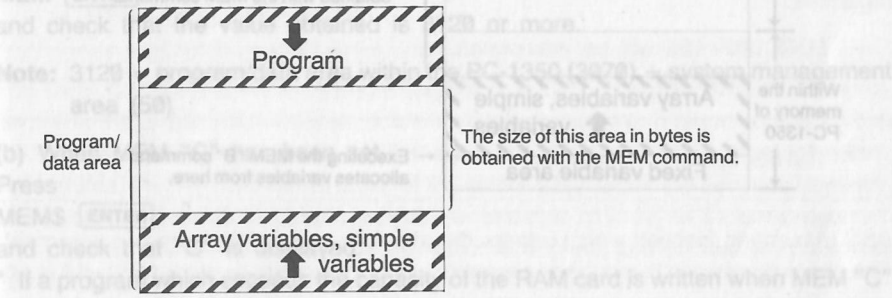
INKEY$ is a string pseudovariable which gives the specified variable the value of the key pressed while the INKEY$ function is executed. [ENTER] , [CLS] , [SHIFT] , [DEF] , [MODE] , [INS] , [DEL] , [SML] , [↑] , [↓] , [►] , and [◄] all have a value of null. INKEY$ is used to respond to the pressing of individual keys without waiting for the ENTER key to end the input.

```
10 A$=INKEY$
20 B=ASC A$
30 IF B=0 THEN GOTO 10
40 IF B ...
```

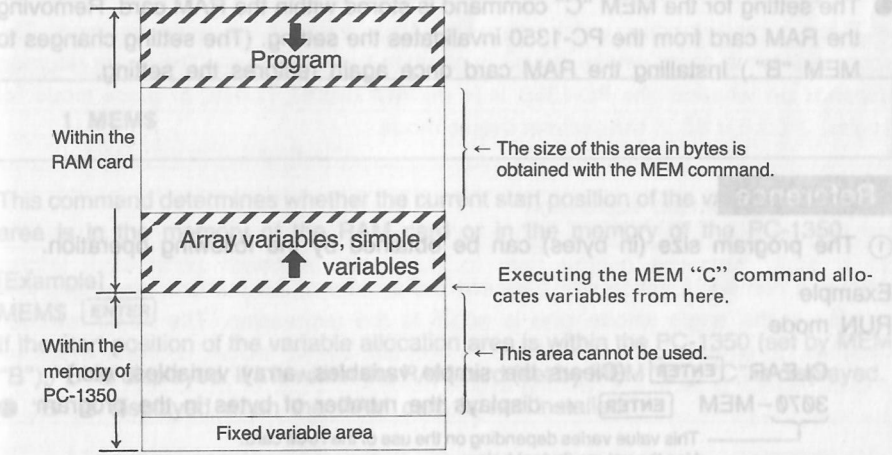
Lines 40 and beyond contain tests for the key and the actions to be taken. (For example: 40 PRINT A$)

1 **MEM**
1 **MEM** "C"
2 **MEM** "B"

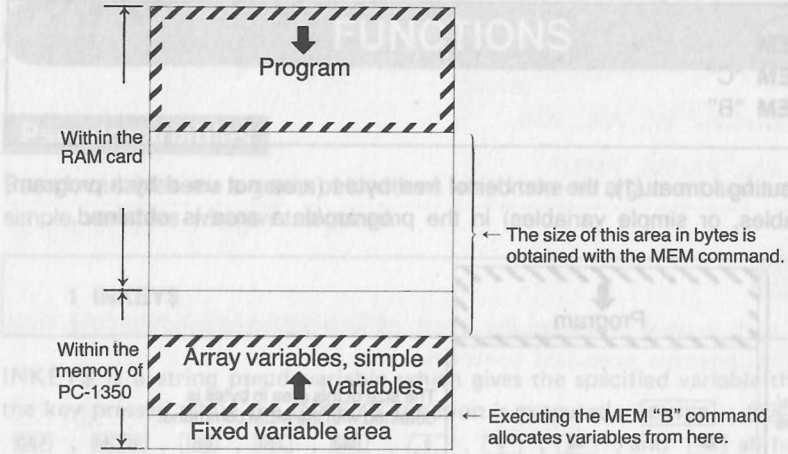
When executing format (1), the number of free bytes (area not used by a program, array variables, or simple variables) in the program/data area is obtained.



When the optional RAM card has been mounted, format (2) can be specified so that variables are allocated on the RAM card.



Format (3) restores the start position of the variable allocation area which was set in the memory of the RAM card by format (2) to the memory in the PC-1350.


Program
Within the RAM card
← The size of this area in bytes is obtained with the MEM command.
Within the memory of PC-1350
Array variables, simple variables
Fixed variable area
← Executing the MEM "B" command allocates variables from here.

- If a RAM card is not mounted, execution of MEM "C" results in an error (ERROR 1).
- The setting for the MEM "C" command is stored within the RAM card. Removing the RAM card from the PC-1350 invalidates the setting. (The setting changes to MEM "B".) Installing the RAM card once again restores the setting.

Reference

① The program size (in bytes) can be obtained by the following operation.

Example
RUN mode

CLEAR [ENTER] (Clears the simple variables, array variables, etc.)
3070–MEM [ENTER] ← displays the number of bytes in the program

This value varies depending on the use of the RAM card.
Use the values (bytes) below.

Capacity of Program/Data Area (bytes)

PC-1350 only	PC-1350 + CE-201M	within CE-201M only	PC-1350 + CE-202M	within CE-202M only
3070	11262	8142*	19454	16334*

* Capacities within the RAM card when MEM "C" is set.

② Checking that the program is stored within the RAM card.
When a program which exceeds the capacity of the RAM card has been written, it will be destroyed if the RAM card is removed from the PC-1350.
Therefore, use the following method when writing a long program to check that the programs stored in the RAM card.

(a) When MEM "B" has been set.
Press
MEM [ENTER]
and check that the value obtained is 3120 or more.

Note: 3120 = program/data area within the PC-1350 (3070) + system management area (50)

(b) When MEM "C" has been set.
Press
MEM$ [ENTER]
and check that "C" is displayed.

* If a program which exceeds the capacity of the RAM card is written when MEM "C" has been set, MEM "C" is cleared and MEM "B" is set.

If "B" is displayed after MEM$ [ENTER] is displayed, press 3120–MEM [ENTER] and shorten the program by the value (in bytes) obtained.

1 MEM$

This command determines whether the current start position of the variable allocation area is in the memory of the RAM card or in the memory of the PC-1350.

[Example]
MEM$ [ENTER]
If the start position of the variable allocation area is within the PC-1350 (set by MEM "B"), "B" is displayed. If it is within the RAM card (set by MEM "C"), "C" is displayed.

- "B" is displayed when the RAM card is not installed.

1 PI

PI is a numeric pseudovariable which has the value of PI. It is identical to the use of the special PI character (π) on the keyboard. Like other numbers the value of PI is kept to 10 digit accuracy (3.141592654).

Numeric Functions

Numeric functions are a group of mathematical operations which take a single numeric value and return a numeric value. They include trigonometric functions, logarithmic functions, and functions which operate on the integer and sign parts of a number. Many dialects of BASIC require that the argument to a function be enclosed in parentheses. The PC-1350 does not require these parentheses, except when it is necessary to indicate what part of a more complex expression is to be included in the argument.

LOG 100+100 will be interpreted as:
(LOG 100)+100 not LOG (100+100).

1 **ABS** numeric expression

ABS is numeric function which returns the absolute value of the numeric argument. The absolute value is the value of a number without regard to its sign. ABS −10 is 10.

1 **ACS** numeric expression

ACS is a numeric function which returns the arccosine of the numeric argument. The arccosine is the angle whose cosine is equal to the expression. The value returned depends on whether the PC-1350 is in decimal degree, radian, or grade mode for angles. ACS.5 is 60 in the decimal degree mode.

1 **ASN** numeric expression

ASN is a numeric function which returns the arcsine of the numeric argument. The arcsine is the angle whose sine is equal to the expression. The value returned depends on whether the PC-1350 is in decimal degree, radian, or grade mode for angles. ASN.5 is 30 in the decimal degree mode.

1 **ATN** numeric expression

ATN is a numeric function which returns the arctangent of the numeric argument. The arctangent is the angle whose tangent is equal to the expression. The value returned depends on whether the PC-1350 is in decimal degree, radian, or grade mode for angles. ATN 1. is 45 in the decimal degree mode.

1 **COS** numeric expression

COS is a numeric function which returns the cosine of the angle argument. The value returned depends on whether the PC-1350 is in decimal degree, radian, or grade mode for angles. COS 60 is .5 in the decimal degree mode.

1 **DEG** numeric expression

The DEG function converts an angle argument in DMS (Degree, Minute, Second) format to DEG (Decimal Degree) form. In DMS format the integer portion of the number represents the degrees, the first and second digits of the decimal represent the minutes, the third and fourth digits of the decimal represent the seconds, and any further digits represent decimal seconds. For example, 55° 10' 44.5" is represented as 55.10445. In DEG format the integer portion is degrees and the decimal portion is decimal degrees. DEG 55.10445 is 55.17902778.

1 **DMS** numeric expression

DMS is a numeric function which converts an angle argument in DEG format to DMS format (see DEG). DMS 55.17902778 is 55.10445.

1 **EXP** numeric expression

EXP is a numeric function which returns the value of e (2.718281828—the base of the natural logarithms) raised to the value of the numeric argument. EXP 1 is 2.718281828.

1 **INT** numeric expression

INT is a numeric function which returns the integer part of its numeric argument. INT PI is 3.

1 **LN** numeric expression

LN is a numeric function which returns the logarithm to the base e (2.718281828) of its numeric argument. LN 100 is 4.605170186).

1 **LOG** numeric expression

LOG is a numeric function which returns the logarithm to the base 10 of its numeric argument. LOG 100 is 2.

1 **RND** numeric expression

RND is a numeric function which generates random numbers. If the value of the argument is less than 1 but greater than or equal to zero, the random number is less than 1 and greater than or equal to zero. If the argument is an integer greater than or equal to 1, the result is a random number greater than or equal to 1 and less than or equal to the argument. If the argument is greater than or equal to 1 and not an integer, the result is a random number greater than or equal to 1 and less than or equal to the smallest integer which is larger than the argument: (In this case, the generation of the random number changes depending on the value of the decimal portion of the argument.):

	Result	
Argument	Lower Bound	Upper Bound
.5	0<	<1
2	1	2
2.5	1	3

The same sequence of random numbers is normally generated because the same "seed" is used each time the PC-1350 is turned on. To randomize the seed, see the RANDOM verb.

1 **SGN** numeric expression

SGN is a numeric function which returns a value based on the sign of the argument. If the argument is positive, the result is 1; if the argument is zero, the result is 0; if the argument is negative, the result is −1. SGN −5 is −1.

1 **SIN** numeric expression

SIN is a numeric function which returns the sine of the angle argument. The value returned depends on whether the PC-1350 is in decimal degree, radian, or grade mode for angles. SIN 30 is 0.5.

1 **SQR** numeric expression

SQR is a numeric function which returns the square root of its argument. It is identical to the use of the special square root symbol ($\sqrt{\ }$) on the keyboard. SQR 4 is 2.

1 **TAN** numeric expression

TAN is a numeric function which returns the tangent of its angle argument. The value returned depends on whether the PC-1350 is in decimal degree, radian, or grade mode for angles. TAN 45 is 1.

String Functions

String functions are a group of operations used for manipulating strings. Some take a string argument and return a numeric value. Some take a string argument and return a string. Some take numeric value and return a string. Some take a string argument and one or two numeric arguments and return a string. Many dialects of BASIC require the argument of a function to be enclosed in parentheses. The PC-1350 does not require these parentheses, except when it is necessary to indicate what part of a more complex expression is to be included in the argument. String functions with two or three arguments all require the parentheses.

1 **ASC** string expression

ASC is a string function which returns the numeric character code value of the first character in its argument. The chart of character codes and their relationship to characters is given in Appendix B. ASC "A" is 65.
The PC-1350 uses ASCII codes and their characters.

1 **CHR$** numeric expression

CHR$ is a string function which returns the character which corresponds to the numeric character code of its argument. The chart of character codes and their relation- ship to characters is given in Appendix B. CHR$ 65 is "A".

Note: If characeter code 13 is specified when manually executing the CHR$ command, the specified contents that follow it will not be displayed.

[Examples]
CHR$70+CHR$71+CHR$13+CHR$75+CHR$76
[ENTER] → FG
Characters K and L for codes 75 and 76 are not displayed.

1 **LEFT$** (string expression, numeric expression)

LEFT$ is a string function which returns the leftmost part of the string first argument. The number of characters returned is determined by the numeric expression. LEFT$ ("ABCDEF", 2) is "AB".

1 **LEN** string expression

LEN is a string function which returns the length of the string argument.
LEN "ABCDEF" is 6.

1 **MID$** (string expression, num. exp. 1, num. exp. 2)

MID$ is a string function which returns a middle portion of the string in the first argument. The first numeric argument indicates the first character position to be included in the result. The second numeric argument indicates the number of characters that are to be included. MID$ ("ABCDEF", 2,3) is "BCD".

1 **RIGHT$** (string expression, numeric expression)

RIGHT$ is a string function which returns the rightmost part of the string first argument. The number of characters returned is determined by the numeric argument. RIGHT$ ("ABCDEF", 3) is 'DEF'.

1 **STR$** numeric expression

STR$ is a string function which returns a string which is the character representation of its numeric argument. It is the reverse of VAL. STR$ 1.59 is '1.59'.

1 **VAL** string expression

VAL is a string function which returns the numeric value of its string argument. It is the reverse of STR$. The VAL of a non-number is zero. VAL "1.59" is 1.59.

Note: The character-string convertible by VAL function to a numerical value consists of numerals (0 to 9), symbols (+ and −) and a symbol (E) indicating an exponential portion. No other characters and symbols are included. If a character-string includes other characters and symbols, any character-string on the right of that character-string will be ignored. If included in a character-string, a space is usually regarded as non-existing.

1 **GCURSOR** (expression 1, expression 2)

Abbreviation: GC., GCU., GCUR., GCURS., GCURSO.

See also: CURSOR, GPRINT

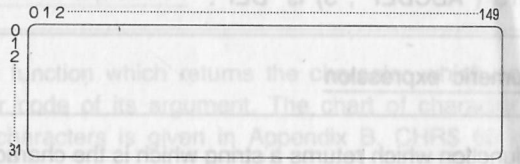
Purpose

Specifies the display start position in dot (point) units.

Use

The GCURSOR command specifies the start position of the display for the GPRINT command. (Moves the graphic cursor to the specified position.)

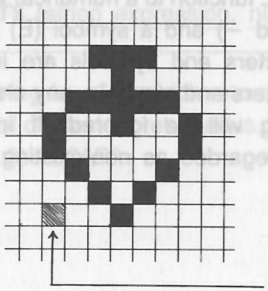
* The display unit is composed of 150 horizontal and 32 vertical dots (points). Each dot is assigned a number ranging 0–149 in the X direction and 0–31 in the Y direction. The display start position is specified by specifying the number in the X direction with expression 1 in the format above and the number in the Y direction with expression 2.



Example

```
10 GCURSOR (71, 20)
20 GPRINT "1824458F452418"
```

Executing this program displays the following near the center of the display unit. (The shaded part is not displayed.)



In the program above, this shaded position is the display start position (71, 20).

Note: The values of expression 1 and expression 2 can be specified in the range of −32768 to +32767. However, if the values for expression 1 and expression 2 exceed the ranges of 0–149 and 0–31, respectively, the specified position will be outside the boundaries of the display unit and a vertical position (position which does not actually exist) will be specified for the display start position.

* The display start position of (0, 7) will be specified when the RUN command or CLS command is executed or when [SHIFT] and [CA] are pressed. When the program was started using the GOTO command or defined key, the value for the y-direction is retained. The x-direction returns to 0.

1 **GPRINT** string
2 **GPRINT** expression; expression; expression;...
3 **GPRINT**

Abbreviation: GP., GPR., GPRI., GPRIN.

See also: GCURSOR, PRINT

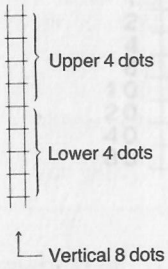
Purpose

Displays the specified dot pattern.

Use

The GPRINT command displays the specified dot pattern. A vertical line of 8 dots is specified as one dot pattern.

* In format (1), the 8-dot pattern, divided into a lower 4 dots and an upper 4 dots, is specified by a string enclosed within " ," where the 4-dot patterns are represented by hexadecimal numbers.



Hexadecimal number	Dot pattern	Hexadecimal number	Dot pattern	Hexadecimal number	Dot pattern	Hexadecimal number	Dot pattern
0		4		8		C	
1		5		9		D	
2		6		A		E	
3		7		B		F	

GPRINT "○○ ○○ ○○ ..."

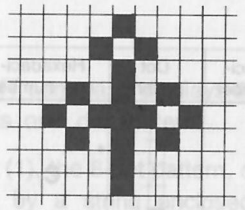
A single group (pair) of 2 numerals specifies one vertical (8-dot) pattern. Any numeral remaining at the end of the string which is not in a pair will be ignored.

Example

GPRINT "102812FD122810"

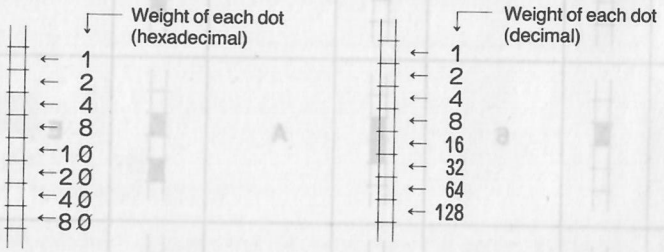
Represents the upper (4-dot) pattern.

Represents the lower (4-dot) pattern.



0 8 2 D 2 8 0 ← Upper (4-dot) pattern represented in hexadecimal.
1 2 1 F 1 2 1 ← Lower (4-dot) pattern represented in hexadecimal.

* In format (2), a vertical 8-dot pattern is specified as a hexadecimal or decimal value. A "weight" is assigned to each dot in the vertical 8-dot pattern as shown below.

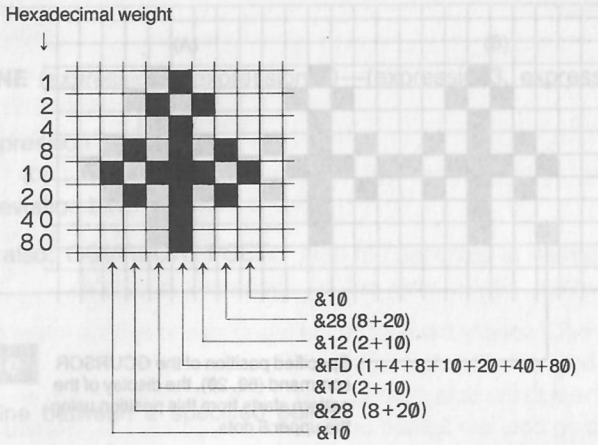


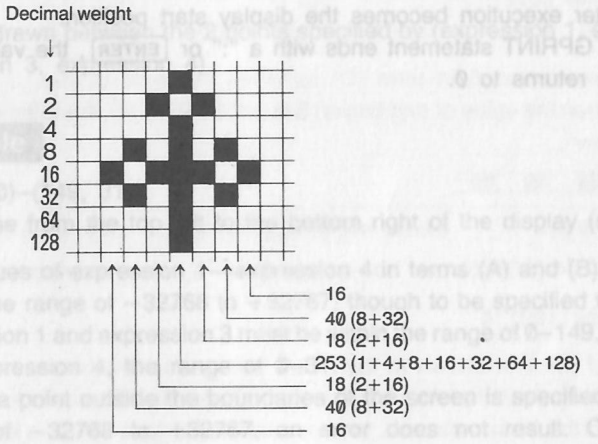
Specify the dot pattern with a numeric value with the sum of the weights of the dots to be lit on the display unit.

Example

* Specifying dot patterns in hexadecimal.
  GPRINT &10; &28; &12; &FD; &12; &28; &10

* Specifying dot patterns in decimal.
  GPRINT 16; 40; 18; 253; 18; 40; 16



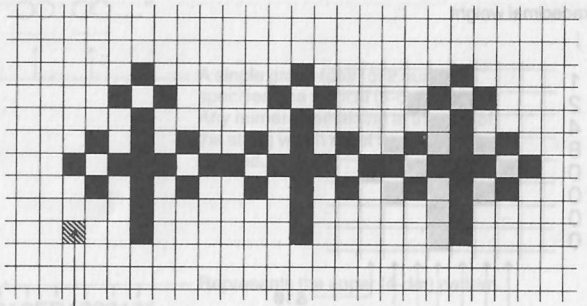


In format (3), the previous graphic display continues to be displayed.

* When the display start position has been specified with the GCURSOR command, the first dot pattern (out of the entire pattern displayed with the GPRINT command) is displayed using the upper 8 dots which includes the specified display start position (dot).

Example

```
10 AA$="102812FD122810"
20 GCURSOR (60, 20)
30 GPRINT AA$;AA$;AA$
```



Specified position of the GCURSOR command (60, 20). the display of the pattern starts from this position using the upper 8 dots.

Note: When the GPRINT statement ends with a ";", the column to the right of the display after execution becomes the display start position.
When the GPRINT statement ends with a ":" or [ENTER], the value for the x-direction returns to 0.

1 **LINE** (expression1, expression 2)—(expression3, expression 4), {S / R / X}, expression 5, {B / BF}

(A) (B) (C) (D) (E)

Abbreviation:LIN.

See also: GCURSOR, PSET

Purpose

Draws a line between 2 specified points.

Use

A line is drawn between the 2 points specified by (expression 1, expression 2) and (expression 3, expression 4).

Example

LINE (0, 0)–(149, 31)

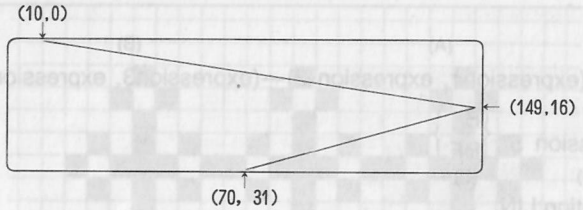
A line from the top left to the bottom right of the display (screen) is drawn.

* The values of expression 1—expression 4 in terms (A) and (B) can be specified within the range of −32768 to +32767, though to be specified within the screen, expression 1 and expression 3 must be within the range of 0–149, and expression 2 and expression 4, the range of 0–31.
Even if a point outside the boundaries of the screen is specified but is within the range of −32768 to +32767, an error does not result. Only the portions corresponding to the area within the boundaries of the screen are displayed. Specifying a point beyond the range results in an error (ERROR 3).

* Term (A) (expression 1, expression 2) can be omitted. If omitted, the line is drawn from either position (0, 0) or the position specified by term (B) (expression 3, expression 4) in a LINE command executed directly before.

Example

```
 5 CLS : WAIT 0
10 LINE (10, 0)–(149, 16)
20 WAIT : LINE–(70, 31)
```



Note: Since the screen is composed of dots, diagonal lines may not appear to be straight. Further, curves may not be accurately displayed.

* S, R, or X interm (C) specify the dots on the line drawn to light up, clear or reverse.

S: Draws the line with the dots lighted. (Sets the dots.)

R: Draws the line with the dots cleared. This is used to draw a line in an area where the surrounding dots are lighted or to clear an existing line. (Resets the dots.)

X: Draws the line and clears the dots if already lit or lights the dots if not lit. (Reverses the dots.)

If none of S, R, or X is specified, S is assumed.

* The value of expression 5 in term (D) specifies the type of line.
For example, when the value of expression 5 is 26214 (&6666), the following type of line will be drawn.

16 dots — A line is drawn by repeating the pattern shown on the left.

The number 26214 (&6666) can be expressed as a binary number as follows:

0110011001100110

If the 16 dots of the line shown in the figure above and the binary number are compared, it can be seen that the dots corresponding to the 1's are lit and the dots corresponding to the 0's are cleared. In this manner, the type of line is specified by the 0's and 1's after converting the value of expression 5 into a 16 digit binary number. Therefore, the line does not appear on the screen when the value of expression 5 is 0 and a solid line appears when the value is 65535 (&FFFF). A solid line is also displayed if expression 5 is omitted. However, if R is specified in term (C), the opposite occurs, and if X is specified, the dots corresponding to the 1's are reversed.

* The value of expression 5 can be specified in the range of 0-65535 (&FFFF).

* Term (E) draws a square, the diagonal of which is a line connecting 2 points specified by terms (A) and (B).

B: Draws a square.

BF: Draws a square filled in with lines.

Examples

```
10 CLS : WAIT 0
20 AA$="102812FD122810"
30 GCURSOR (64, 20)
40 GPRINT AA$; AA$; AA$
50 LINE (24, 0)-(124, 31), &F18F, B
60 LINE (34, 3)-(114, 28), X, BF
70 GOTO 60
```

1 **POINT** (expression 1, expression 2)

Abbreviation: POI., POIN.

See also: GCURSOR, PSET, PRESET

Purpose

Reads the state of the specified dot.

Use

If the dot specified by (expression 1, expression 2) is lit, a "1" is returned, and if cleared, a "0" is returned.

If the specified dot lies beyond the boundaries of the screen, a "−1" is returned.

* The values of expression 1 and expression 2 can be specified within the range of −32768 to +32767. However, dots on the screen are in the range of 0–149 for expression 1 and 0–31 for expression 2.

Example

```
10 CLS : WAIT 0:A=75
20 LINE (50, 0)-(50, 31)      ←┐ Draws 2 vertical lines.
30 LINE (100, 0)-(100, 31)    ←┘
40 PSET (A, 16)               ← Lights up a dot (point) between the 2 lines.
50 B= POINT (A+1, 16)         ← Checks to see whether the next dot on the right is lit.
60 IF B THEN 150              ← If lit, jump to line 150.
70 PSET (A+1, 16)             ← If cleared, light it.
80 PRESET (A, 16)             ← Then, clear the dot lit up earlier.
90 A=A+1                      ← Move one dot to the right.
100 GOTO 50                   ← Go back to line 50.
150 B=POINT (A−1, 16)         ← Checks to see whether the dot to the left of the lit dot is lit.
160 IF B THEN 50              ← If lit, go to line 50.
170 PSET (A−1, 16)            ← If cleared, light it.
180 PRESET (A, 16)            ← Then, clear the dot lit up earlier.
190 A=A−1                     ← Move one dot to the left.
200 GOTO 150                  ← Go back to line 150.
```

Executing this program moves a dot back and forth between 2 vertical lines drawn on the screen.

1 **PRESET** (expression 1, expression 2)

Abbreviation: PRE., PRES., PRESE.

See also: PSET, GCURSOR, POINT

Purpose

Clears (resets) the specified dot on the screen.

Use

Clears the dot specified by (expression 1, expression 2).

* The values of expression 1 and expression 2 can be specified within the range of −32768 to +32767. However, dots on the screen are in the range of 0–149 for expression 1 and 0–31 for expression 2.

Example

```
10 CLS : WAIT 0
20 LINE (20, 0)-(130, 31), BF
30 FOR X=-25 TO 25 STEP 0.5
40 Y=-1* SQR ABS (25*25-X*X)
50 PRESET (X+75, Y+31)
60 NEXT X
70 WAIT : GPRINT
```

Executing this program draws a semicircle inside a filled square.

1 **PSET** (expression 1, expression 2)
2 **PSET** (expression 1, expression 2), X

Abbreviation: PS., PSE.

See also: PRESET, GCURSOR, POINT

Purpose

Lights up or reverses the specified dot on the screen.

Use

* Foirmat (1) lights up the dot specified by (expression 1, expression 2).
* Format (2) clears the dot specified by (expression 1, expression 2) if lit and lights it if cleared.
* The values of expression 1 and expression 2 can be specified within the range of −32768 to +32767. However, dots on the screen are in the range of 0–149 for expression 1 and 0–31 for expression 2.

Example

```
10 CLS : WAIT 0 : DEGREE
20 FOR A=0 TO 600
30 B=−1* SIN A
40 Y= INT (B*16) +16
50 X= INT (A/4)
60 PSET (X, Y)
70 NEXT A
80 WAIT : GPRINT
```

Executing this program draws a sine curve on the screen.

1 **CLOSE**

Abbreviation: CLOS.

See also: OPEN

Purpose

Closes the circuit of the serial I/O interface.

Use

This command closes the circuit (in the software sense) of the serial I/O interface which was opened by the OPEN command.
Therefore, after this command is executed, any output to the serial I/O terminal or input from the same terminal cannot be performed.

1 **CONSOLE** expression

Abbreviation: CONS., CONSO., CONSOL.

See also: OPEN, LPRINT, LLIST

Purpose

Sets the number of columns per line for data sending.

Use

This command sets the number of columns per line for data sent from the serial I/O interface (terminal) using the LPRINT or LLIST command.
The PC-1350 sends an end code (CR, LF, or CR + LF) after sending the preset line of data.

* Valid values of the expression are integer in the range of 1–160. If the value of the expression exceeds 160, 160 columns per line will be set. An error (ERROR 3) results if the value is 0 or negative.
  if an expression is not specified, the command is ignored and the number of columns previously set is retained.
* The number of columns becomes 40 when the batteries are replaced or when the reset switch is pressed.

1 **INPUT#1** variable, variable, variable...

Abbreviation: I.#1, IN.#1, INP.#1, INPU.#1

See also: OPEN, LPRINT#1

Purpose

Assigns data, input through the serial I/O interface (terminal), to the specified variables.

Use

* This command is valid only when the circuit of the serial I/O interface is open (after the OPEN command is executed) and is ignored otherwise.
* The INPUT#1 command assigns data (sent in the form as described in the section on the PRINT#1 command) to the specified variables.
  Therefore, the variables are specified as in the PRINT#1 command.

Example

INPUT#1A, AB, C$, E(*)

Data input through the I/O interface is assigned to variables A, AB, and C$, and array variable E( ).

* Be sure that the type of both the specified variables and the input data match (i.e. character or numeric types).
  In the ASCII code system, if a character is assigned to a numeric variable, its value becomes 0. If a number is assigned to a character variable, its contents become a character string. Therefore, if the type of both the specified variable and the input data do not match, unexpected values may result.

  Even if data in a form such as function "SIN 30" is given to a numeric variable, it is assumed to be a character string with its content 0.
  For data in the form of "10 +40", the characters (numbers) after the operator symbol are ignored. Therefore, the data in this case is "10".

Note 1: If CR (control code: 0DH) or NULL (00H) is included within the input data, all data following it may be ignored.

Note 2: Simple variables and array variables must be allocated in the program/data area before executing an INPUT#1 command. An error will result if these variables are not allocated.

1 **LLIST**

2 **LLIST** {expression / "lable"}

3 **LLIST** expression 1, expression 2

Abbreviation: LL., LLI., LLIS.

See also: OPEN, CONSOLE

Purpose

Sends the program contents out of the serial I/O interface (terminal).

Use

The LLIST command is valid under manual operation in the PRO or RUN mode. When the circuit of the serial I/O interface is open due to the OPEN command, the program is sent out in ASCII code.
When the circuit is closed, the program is printed on the printer. (See page 99.)

* In format (1), all programs in the PC-1350 are sent out.

[Example]
When the program below is in the PC-1350, pressing

LLIST [ENTER]

sends out the program in the form shown below.

```
10: OPEN
100: REM **ABC-12**
65279: END
```

Space	1	0	:	O	P	E	N	Space	CR
1	0	0	:	R	E	M	Space	Space	*
*	A	B	C	–	1	2	*	*	CR
6	5	2	7	9	:	E	N	D	Space
CR									

Note: CR is an end code. It is either LF or CR + LF depending on the setting of the OPEN command.

* In format (2), the line indicated by the value of the expression or the line with the specified label is sent out.

* In format (3), the program, from the line indicated by the value of expression 1 to the line indicated by the value of expression 2, is sent out. (Labels can also be used for expression 1 and expression 2.)
Expression 1 or expression 2 can be omitted in format (3).

- If expression 1 is omitted, the program, from the first line to the line indicated by the value of expression 2, is sent out.
- If expression 2 is omitted, the program, from the line indicated by the value of expression 1 to the last line, is sent out.

* If a line corresponding to the value of expression, expression 1 or expression 2 does not exist, the line with the next largest number which does exist will be specified.
An error results (ERROR 1) if the lines specified in expression 1 and expression 2 are the same.

* The LLIST command is ignored if a password has been set.

* If programs have been merged using the MERGE command, the LLIST command functions only for the last merged program.
To list the previously stored programs, execute

LLIST "label",

* The number of print columns per line is set by the CONSOLE command. If set to 23 columns or less, executing the LLIST command results in an error (ERROR 3).

1 **LOAD**

Abbreviation: LOA.

See also: OPEN, CLOAD

Purpose

Loads the data sent from the serial I/O interface (terminal) into the program/data area.

Use

The LOAD command is valid when the circuit of the serial I/O interface is open due to the OPEN command. It is ignored when the circuit is closed.

* Data through the serial I/O interface is read until the end code is reached. Data until the end code is considered to be the first line of the program. The PC-1350 converts the data into a form which can be stored as a program and then transfers (writes) it to the program/data area. Then, data is again read from the serial I/O interface, converted in the same way, and then written into the memory. This operation continues until the text end code (see OPEN command) is read.
* Up to 256 bytes of data can be read at a time. Therefore, if more than 256 bytes of data are sent before the end code is read, an error results.
* The data which has been read is converted and then written to the program/data area. If one line, including the line number, exceeds 80 bytes, an error results. Further, an error also results if the beginning of the line is not a numeric value (line number).
* During execution of the LOAD command, the order of the lines is not rearranged (e.g. ascending order of the line numbers).

Notes:

- Execution of the LOAD command ends when the text end code is read (from the sending side).
  Even if the sending side sends out the entire program, the PC-1350 does not end execution as long as the text end code is not read. In this case, end the execution as follows:
  (1) After the sending side sends the program, have it also send only the text end code.
  (2) Or, press the BRK key to end execution.
- The reserved contents cannot be read from the serial I/O interface.

1 **LPRINT** {expression / character string}

2 **LPRINT** {expression / character string} , {expression / character string} ... , {expression / character string}

3 **LPRINT** {expression / character string} ; {expression / character string} ... ; {expression / character string}

4 **LPRINT** ... {expression / character string} ;
(Format where a ";" is added to the end of 1 and 3 above.)

5 **LPRINT**

Abbreviation: LP., LPR., LDRI., LPRIN.

See also: OPEN, CONSOLE, USING

Purpose

Sends the specified information out through the serial I/O interface (terminal).

Use

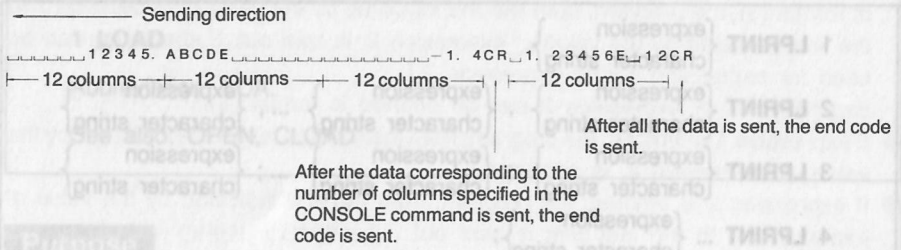
When the circuit of the serial I/O interface is opened by the OPEN command, the specified information is sent out through the serial I/O terminal in ASCII code. When the circuit is closed, LPRINT prints information on the printer. (See page 133.)

* In format (1), the value of the expression or the character string is sent from its beginning.
  If the value of the expression is negative a "−" sign is sent before the value. If positive, a space is sent.
* In format (2), 12 column (digit) divisions are automatically set. The value of a single expression or a character string is sent within a range of 12 columns (digits).

[Example]

```
10 OPEN "1200, N, 8, 1, A, C"
20 CONSOLE 36
30 LPRINT 12345, "ABCDE",
   −7/5, 1. 23456789E12
```

Executing this program sends information in the following form.


Sending direction
␣␣␣␣␣12345. ABCDE␣␣␣␣␣

1 **OPEN** "baud rate, parity, word length, stop bit, type of code, end code, text end code"

2 **OPEN**

Abbreviation: OP., OPE.

See also: CLOSE

Purpose

Allows data to be transferred through the I/O interface. Also sets the I/O conditions.

Use

Format (1) enables data to be transferred through the I/O interface (serial I/O terminal). It also sets the conditions for the data transfer with the connected equipment. The conditions are specified in the following form:
"baud rate, parity, word length, stop bit, type of code, end code, text end code"

Baud Rate: 300, 600, 1200
Specifies the modulation rate (transfer rate). For the PC-1350, 300 baud, 600 baud, or 1200 baud can be selected.
(1 baud = 1 bit/sec)

Parity: N, E, O
Specifies the type of parity by a character.
N: No parity bit is transmitted nor received.
E: Specifies even parity.
O: Specifies odd parity.

Word Length: 7, 8
Specifies how many bits to be transmitted or received per character. Either 7 or 8 bits can be specified.

Number of Stop Bits: 1, 2

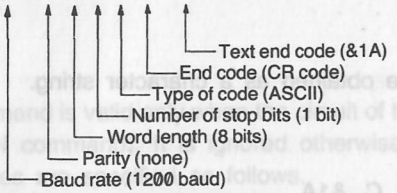
Type of Code: A
Only ASCII codes can be transmitted or received. Therefore, A is always specified.

End Code: C, F, L
Specifies the type of end code to indicate the end of data (delimiting), end of a program line, etc.
C: Specifies the CR (carriage return) code.
F: Specifies the LF (line feed) code.
L: Specifies the CR code + LF code.

Text End Code: &00–&FF
Specifies the text end code to indicate the end of the program, etc.
(May be required when using the SAVE or LOAD commands.)

Example

```
OPEN "1200,N,8,1,A,C,&1A"
```


Text end code (&1A)
End code (CR code)
Type of code (ASCII)
Number of stop bits (1 bit)
Word length (8 bits)
Parity (none)
Baud rate (1200 baud)

The conditions in the example above are set after the batteries are replaced or after the reset switch is pressed.

* Any condition specified in the OPEN command can be omitted. If omitted, the current condition remains unchanged.

Example

```
OPEN ",,,2"
```

Only the number of stop bits is changed.

* In format (2), all conditions set previously are retained. This format enables data to be transferred through the I/O interface.

* Executing the OPEN command while the circuit of the I/O interface is open and ready for data transfer (due to prior execution of the OPEN command) results in an error (ERROR 8).
Execute the CLOSE command to close the circuit. (The circuit also closes when the RUN command is executed, when the program ends, or when the power is switched off.)
The previously set conditions are retained even after the CLOSE command is executed.

1 **OPEN$**

Abbreviation: OP.$, OPE.$

See also: OPEN

Purpose

Obtains the currently set I/O conditions.

Use

The currently set I/O conditions are obtained as a character string.

Example

OPEN$ [ENTER] 1200, N,8, 1, A, C, &1A

1 **PRINT#1** variable, variable, variable ...

Abbreviation: P.#1, PR.#1, PRI.#1, PRIN.#1

See also: OPEN, INPUT#1

Purpose

Sends the contents of the specified variables through the serial I/O interface (terminal).

Use

This command is valid only when the circuit of the serial I/O interface is open (due to the OPEN command). It is ignored otherwise.

* Variables are specified as follows.

Fixed Variables: Specify each variable name.
[Example] A, B, C$
Note: Fixed variables cannot be specified in the form of A*

Simple Variables: Specify each variable name.
[Example] AA, B1$, C2

Array Variables: Specify in the form of array name (*).
[Example] B(*), C$(*)
Specified in this manner, the contents of all elements in the array are sent. (Array elements cannot be specified individually).
[Example] 50 PRINT #1A, AB, C$, E(*)

* When data is sent, the end code is added to the end of the contents of each variable. The end code is added to the end of the contents of each element for array variables also.

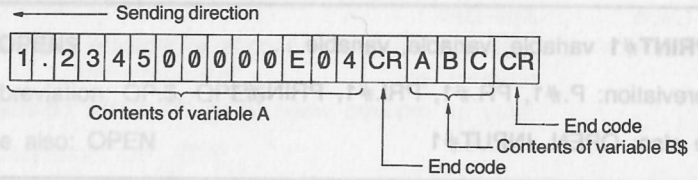
[Example] When

A = 12345 and B$ = "ABC"

executing

PRINT#1A,B$

Sends A and B$ in the following form. (Provided the end code is CR.)



* If the value of the numeric variable is negative, a "−" sign is sent before the value.

* The elements of an array are sent in the following sequence.
1 dimensional array

[Example] for B (3)

B (0) → B (1) → B (2) → B (3)

2 dimensional array

[Example] For C (2, 3)

C (0, 0) → C (0, 1) → C (0, 2) .....

Notes:

* The locations for extended variables such as A(27) and beyond, simple variables, and/or array variables must be allocated in the program/data area before the PRINT#1 command is executed. An error results if an attempt is made to send the contents of a variable which has not been allocated.
An error also results if the type of the specified variable (numeric or character) and that of within the PC-1350 do not match.

* If symbols such as "π" and "√‾" are included in the data to be sent, they are converted to "PI" and "SQR," respectively, and then sent.

1 **SAVE**

Abbreviation: SA., SAV.

See also: OPEN, LLIST

Purpose

Sends the program within the PC-1350 out through the serial I/O interface (terminal).

Use

When the circuit of the serial I/O interface is open due to the OPEN command, the program is sent in ASCII code.
The command is ignored if the circuit is closed.

* After the entire program is sent, the text end code is sent.

* The SAVE command is ignored if a password has been set.

1 **BASIC**

Abbreviation: BA., BAS., BASI.

See also: TEXT

Purpose

Clears the text mode.
(valid only in manual operation in the program mode)

Use

Executing this command clears the text mode and returns the mode to BASIC. As the mode returns to BASIC, the prompt symbol changes from "<" to ">".

- Changing from the text mode to the BASIC mode usually changes the text in the PC-1350 to a program (internal code).
  However, abbreviations such as "P." and "I." are not converted to their respective commands. (In this case, call the cursor to the line and press the [ENTER] key to convert it to a command.)
  Due to characteristics of the text function, commands and formats included in the text but not found in the PC-1350 may not be executable.
- During program conversion, the "*" mark is displayed on the right end of the 4th line of the display unit.
- If a password has been set, executing the BASIC command results in an error (ERROR 1).

1 **TEXT**

Abbreviation: TE., TEX.

See also: BASIC

Purpose

Sets the text mode.
(valid only in manual operation in the program mode)

Use

The text function is used when inputting a program written for a higher level personal computer. The program input by the PC-1350 is sent to the personal computer through the serial I/O interface.

- Executing the TEXT command sets the text mode. In the text mode, a number corresponding to the line number and then information corresponding to program commands or data are entered. Then the [ENTER] key is pressed to write the input to the program/data area.
  However, the written contents, unlike in the BASIC mode, are not converted to commands (internal codes). The text is stored as they are (as characters and/or numbers) in ASCII codes. The text is arranged in the order of the numbers corresponding to the line number at the beginning of each line. (Line number editing function.)
- The text written in the text mode is stored as it is. Therefore, command abbreviations in BASIC (such as I. for INPUT) are displayed and stored as they are.
- If a program is stored in the internal code of the PC-1350 with the text mode set, it is converted to ASCII code.
- During program conversion, the "*" mark is displayed on the right end of the 4th line of the display unit.
- The prompt symbol is "<" in the text mode. (It is usually ">".)
- One line in the text mode (including line number and [ENTER]) must not exceed 80 characters (80 bytes). If a line exceeds 80 characters due to program conversion, the excess part will be cleared.

80 bytes (end is [ENTER])

```
10:PRINT"ABC..... 1234567890"
```

Converted to ASCII code.

```
10:PRINT  "ABC..... 123456
```

80 characters (end is [ENTER])

In this example, the PRINT command is one byte in internal code but takes up 6 bytes in ASCII code. Because of this, the last few characters are deleted (7890).

- The number of bytes increases when converting a program from internal code to ASCII code, as shown in the example. If, as a result, the capacity of the program area is exceeded, the program converted up to that point is converted back to internal code and an error will result (ERROR 6).
- If a password has been set, an error (ERROR 1) occurs when the TEXT command is executed.

CHAPTER 10
TROUBLESHOOTING

This chapter provides you with some hints on what to do when your SHARP PC-1350 does not do what you expect it to do. It is divided into two parts—the first part deals with general machine operation and the second with BASIC programming. For each problem there are a series of suggestions provided. You should try each of these, one at a time, until you have fixed the problem.

Machine Operation

If:	Then You Should:
You turn on the machine but there is nothing on the display	1. Check to see that the slide power switch is set to ON position. 2. Push the [ON BRK] key to see if AUTO POWER OFF has been activated. 3. Replace the batteries. 4. Adjust the contrast control.
There is a display, but no response to keystrokes	1. Press [CLS] key to clear. 2. Press [CA] ([SHIFT] [CLS]) to clear. 3. Turn OFF and ON again. 4. Hold down any key and push RESET. 5. Push RESET without any key.
You have typed in a calculation or answer and get no response	1. Push [ENTER].
You are running a BASIC program and it displays something, and stops	1. Push [ENTER].
You enter a calculation and it is displayed in BASIC statement format (colon after the first number)	1. Change from the PROgram into the RUN mode for calculations.
You get no response from any keys.	1. Hold down any key and push RESET. 2. If you get no response from any key even when the above operation is performed, push the RESET without pushing any key. This will clear the program, data and all reserved contents.

BASIC Debugging

When entering a new BASIC program, it is usual for it not to work the first time. Even if you are simply keying in a program that you know is correct, such as those provided in this manual, it is usual to make at least one typing error. If it is a new program of any length, it will probably contain at least one logic error as well. Following are some general hints on how to find and correct your errors.

You run your program and get an error message:

1. Go back to the PROgram mode and use the [↑] or the [↓] keys to recall the line with the error. The cursor will be positioned at the place in the line where the PC-1350 got confused.
2. If you can't find an obvious error in the way in which the line is written, the problem may lie with the values which are being used. For example, CHR$(A) will produce a space if A has a value of 1. Check the values of the variables in either the RUN or the PROgram mode by typing in the name of the variable followed by [ENTER].

You RUN the program and don't get an error message, but it doesn't do what you expect.

3. Check through the program line by line using LIST and the [↓] and [↑] keys to see if you have entered the program correctly. It is surprising how many errors can be fixed by just taking another look at the program.
4. Think about each line as you go through the program as if you were the computer. Take sample values and try to apply the operation in each line to see if you get the result that you expected.
5. Insert one or more extra PRINT statements in your program to display key values and key locations. Use these to isolate the parts of the program that are working correctly and the location of the error. This approach is also useful for determining which parts of a program have been executed. You can also use STOP to temporarily halt execution at critical points so that several variables can be examined.
6. Use TRON and TROFF, either as commands or directly within the program to trace the flow of the program through individual lines. Stop to examine the contents of critical variables at crucial points. This is a very slow way to find a problem, but sometimes it is also the only way.

To continue the program, press the [↓] (Down Arrow) key once. This causes the next line to be executed and its line number to be displayed. Again, you may review the line with the Up Arrow key. You may also check the contents of any variable by typing its name and pressing [ENTER] :

[A] [ENTER] (when A=4 is input before [A] [ENTER] operation)

```
                    4
```

It is necessary to press the [↓] (Down Arrow) key once for each line to be executed until the program ends. If you do not wish to continue normal line-by-line execution, press the [ENTER] key to suspend execution of the program. If you change your mind again, suspended programs may be continued with the CONT command.

A sample session, using our hypotenuse program, follows:

Input	Display
	>
[T] [R] [O] [N]	TRON_
[ENTER]	>
[R] [U] [N]	RUN_
[ENTER]	?
[3]	3_
[ENTER]	?
[4]	4_
[ENTER]	10:
[↑]	10:INPUT A, B
[↓]	20:
[↑]	20:A=A*A : B=B*B
[A]	A_
[ENTER]	9.
[B]	B_
[ENTER]	16.
[↓]	30:
[H]	H_
[ENTER]	5.
[↓]	HYPOTENUSE=5.
[↑]	40:PRINT "HYPOTENUSE="; H
[↓]	40:
[↓]	>

No matter how careful you are, eventually you will create a program which does not do quite what you expect it to . In order to isolate the problem. Sharp's designers have provided a special method of executing programs known as the "Trace" mode. In the Trace mode, the PC-1350 will display the line number of each program line and will halt after the execution of that line. This allows you to follow (or trace) the sequence of instructions as they are actually performed. When the program pauses after the execution of a line, you may inspect or alter the values of variables.

The form of the instruction for initiating the Trace mode is simply: TRON. The TRON instructon may be issued as a command (in RUN mode) or it may be embedded, as a statement, within a program. Used as a command, TRON informs the PC-1350 that tracing is requird during the execution of all subsequent programs. The programs to be traced are then started in a normal manner, with a GOTO or RUN command.

If TRON is used as a statement, it will initiate the Trace mode only when the line containing it is executed. If, for some reason, that line is never reached, the Trace mode will remain inactive.

Once initiated, the Trace mode of operation remains in effect until canceled by a TROFF instruction. The TROFF instruction may also be issued as either a command or a statement. The Trace mode can also be canceled by the key sequence:

[SHIFT] [CA / CLS]

As and example in using the Trace mode, enter the following program to compute the length of the hypotenuse of a triangle given the length of the sides:

Program Listing:

```
10 INPUT A, B
20 A=A*A : B=B*B
30 H=√(A+B)
40 PRINT "HYPOTENUSE= ";H
```

In RUN mode, issue the TRON command, followed by the RUN command. Notice the INPUT command operates in the usual manner by displaying a question mark for each input value required. As soon as you have entered two values, the line number of the INPUT statement appears:

```
10:
```

By pressing the [↑] (Up Arrow) key and holding it, you may review the entire line:

```
10: INPUT A, B
```

* In the trace mode, after the calculated result is displayed at the location specified with the CURSOR command, the next line number is displayed on the following line. (See page 113 for a description on the CURSOR.)
* In the trace mode, if variables are called or if a calculation is performed manually when the display starting position has been specified with the cursor command, the display starting position will be cleared.

CHAPTER 11
MAINTENANCE OF THE PC-1350

To insure trouble-free operation of your SHARP PC-1350 we recommend the following:

* Always handle the pocket computer carefully as the liquid crystal display is made of glass.
* Keep the computer in an area free from extreme temperature changes, moisture, or dust. During warm weather, vehicles left in direct sunlight are subject to high temperature build up. Prolonged exposure to high temperature may cause damage to your computer.
* Use only a soft, dry cloth to clean the computer. Do not use solvents, water, or wet cloths.
* To avoid battery leakage, remove the batteries when the computer will not be in use for an extended period of time.
* If service is required, the computer should only be returned to an authorized SHARP Service Center.
* If the computer is subjected to strong static electricity or external noise it may "hang up" (all keys become inoperative). If this occurs, press the ALL RESET button while holding down any key. (See Troubleshooting).
* Keep this manual for further reference.

APPENDIX A
ERROR MESSAGES

There are nine different codes built into the PC-1350. The following table will explain these codes.

Error Number **Meaning**

1 Syntax error.

● This means that the PC-1350 can't understand what you have entered. Check for things such as semicolons on the ends of PRINT statement, misspelled words, and incorrect usages.

```
3*/2
```

2 Calculation error

Here you have probably done one of three things:

1. Tried to use too large a number.
   Calculation results are greater than 9.999999999E 99.
2. Tried to divide by zero.

```
5/0
```

3. An illogical calculation has been attempted.

```
LN −30 or ASN 1.5
```

3 Illegal Function (DIMension error/Argument error)

● Array variable already exists.
Array specified without first dimensioning it.
Array subscript exceeds size of array specified in DIM statement.

```
DIM B(256)
```

● Illegal function argument. This means that you have tried to make the computer do something that it just can't handle.
The interval that is greater than 65535.

```
WAIT 66000
```

4 Too Large A Line Number

Here you have probably done one of two things:

1. Tried to use a non-existent line number by the GOTO, GOSUB, RUN, LIST or THEN etc.
2. Tried to use too large a line number. The maximum line number is 65279.

5 Next Without A For ...
Subroutine nesting exceeds 10 levels.

FOR loop nesting exceeds 5 levels.
RETURN verb without a GOSUB, NEXT verb without a FOR, or READ verb without a DATA.

Buffer space exceeded.

6 Memory Overflow.
Generally this error happens when you've tried to DIMension an array that is too big for memory. This can also happen when a program becomes too large.
- The reserve content exceeds 144 bytes.

7 PRINT USING error.
This means that you have put an illegal format specifier into a USING statement.

8 I/O device error.
This error can happen when you have the optional printer and/or cassette recorder connected to the PC-1350. This error can also happen when you use the serial input/output. It means that there is a problem with communication between the I/O device and the PC-1350.

9 Other errors.
This code will be displayed whenever the computer has a problem that isn't covered by one of the other eight error codes. One of the most common causes for this error is trying to access data in a variable is one fashion (e.g. A$) while the data was originally stored in the variable in another fashion (e.g.A).

Regarding Input Errors

When executing a program, an error may occur due to input errors of the program. In this case note the following.

Example: When KPRINT is entered instead of LPRINT

```
                10:K PRINT A$
[►]             10 K PRINT A$
L [ENTER]       10:L PRINT A$
                   └─Space
```


When corrected in this manner, the computer does not recognize it as a command. In this sample, erase KPRINT and re-enter LPRINT.

```
                10:K PRINT A$
[►]             10 K PRINT A$
[DEL] [DEL]     10 A$
[INS] ··· [INS] 10 □□□□□□A$
LPRINT [ENTER]  10:LPRINT A$
                   ↑
Space is not needed when recognized as a command.
```

The command can be checked whether entered correctly by using the cursor key.

(Correct input)	(Wrong input)
10: RADIAN	10:RADAN
[►] 10 RADIAN	[►] 10 RADAN
[►] 10 RADIAN_	[►] 10 RADAN
	[►] 10 RADAN

APPENDIX B
CHARACTER CODE CHART

The following chart shows the conversion values for use with CHR$ and ASC. The column shows the first hex character or the first four binary bits, the row shows the second hex character or the second binary bits. The upper left corner of each box contains the decimal number for the character. The lower right shows tne character. If no character is shown then it is an illegal character on the PC-1350.

For examples, the character "A" is a decimal 65 or a hex 41 or binary 01000001. The character '√' is a decimal 252 or a hex FC or a binary 11111100.

Note:
- When using the CE-126P optional printer, be aware that the 92 (&5C), 249 (&F9), and 250 (&FA) character codes for the PC-1350 (displayed characters) and printer (printed characters) are different characters.
- When using the CE-126P optional printer, do not use the code 0 (& 00)

First 4 Bits

Second 4 Bits: Hex / Binary	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	E 1110	F 1111
0 0000	0 NUL	16	32 SPACE	48 0	64 @	80 P	96 ‾	112 p	128	224	240
1 0001	1	17	33 !	49 1	65 A	81 Q	97 a	113 q	129	225	241
2 0010	2	18	34 "	50 2	66 B	82 R	98 b	114 r	130	226	242
3 0011	3	19	35 #	51 3	67 C	83 S	99 c	115 s	131	227	243
4 0100	4	20	36 $	52 4	68 D	84 T	100 d	116 t	132	228	244
5 0101	5	21	37 %	53 5	69 E	85 U	101 e	117 u	133	229	245 ♠
6 0110	6	22	38 &	54 6	70 F	86 V	102 f	118 v	134	230	246 ♥
7 0111	7	23	39 '	55 7	71 G	87 W	103 g	119 w	135	231	247 ♦
8 1000	8	24	40 (	56 8	72 H	88 X	104 h	120 x	136	232	248 ♣
9 1001	9	25	41)	57 9	73 I	89 Y	105 i	121 y	137	233	249 ▮
A 1010	10	26	42 *	58 :	74 J	90 Z	106 j	122 z	138	234	250 □
B 1011	11	27	43 +	59 ;	75 K	91 [	107 k	123 {	139	235	251 π
C 1100	12	28	44 ,	60 <	76 L	92 \	108 l	124 ¦	140	236	252 √
D 1101	13	29	45 –	61 =	77 M	93]	109 m	125 }	141	237	253
E 1110	14	30	46 .	62 >	78 N	94 ^	110 n	126 ~	142	238	254
F 1111	15	31	47 /	63 ?	79 O	95 _	111 o	127	143	239	255

The character codes are represented as follows.

Example

- Code for "*"
  Hexadecimal &2A
  Decimal 42 (32+10)
- Code for "√"
  Hexadecimal &FC
  Decimal 252 (240+12)

Notes:

- Displaying a character using the CHR$ command:
  The character for code 0 (&00) in the table is null. Nothing is displayed.
  Squares in the table where no characters are listed are spaces.
- Printing a character on the CE-126P using the CHR$ command:
  - Do not use code 0 (&00).
  - Squares in the table where no characters are listed are spaces.
  - Codes 249 (&F9) and 250 (&FA) are spaces.
  - The character for code 92 (&5C) is "◣".

APPENDIX C
FORMATTING OUTPUT

It is sometimes important or useful to control the format as well as the content of output. The PC-1350 controls display formats with the USING verb. This verb allows you to specify:

* The number of digits
* The location of the decimal point
* Scientific notation format
* The number of string characters

These different formats are specified with an "output mask." This mask may be a string constant or a string variable:

```
10: USING "####"
20: M$="&&&&&&"
30: USING M$
```

When the USING verb is used with no mask, all special formatting is cancelled.

```
40: USING
```

A USING verb may also be used within a PRINT statement:

```
50: PRINT USING M$; N
```

Wherever a USING verb is used, it will control the format of all output until a new USING verb is encountered.

Numeric Masks

A numeric USING mask may only be used to display numeric values, i.e., numeric constants or numeric variables. If a string constant or variable is displayed while a numeric USING mask is in effect, the mask will be ignored. A value which is to be displayed must always fit within the space provided by the mask. The mask must reserve space for the sign character, even when the number will always be positive. Thus a mask which shows four display positions may only be used to display numbers with three digits.

Specifying Number of Digits

The desired number of digits is specified using the '#' character. Each '#' in the mask reserves space for one digit. The display or print always contains as many characters as are designated in the mask. The number appears to the far right of this field; the remaining positions to the left are filled with spaces. Postive numbers therefore

always have at least one space at the left of the field. Since the PC-1350 maintains a maximum of 10 significant digits, no more than 11 '#' characters should be used in a numeric mask. When the total number of columns of the integer part specified exceed 11, this integer part is regarded as 11 digits in the PC-1350.

Note: In all examples in this appendix the beginning and end of the displayed field will be marked with a 'I' character to show the size of the field.

Statement	Display
10: USING "####"	(Set the PC-1350 to the RUN mode, type RUN, and press [ENTER].)
20: PRINT 25	| 25|
30: PRINT −350	|−350|
40: PRINT 1000	**ERROR 7 IN 40**

Notice that the last statement produced an error because 5 positions (4 digits and a sign space) were required, but only 4 were provided in the mask.

Specifying a Decimal Point

A decimal point character, '.', may be included in a numeric mask to indicate the desired location of the decimal point. If the mask provides more significant decimal digits than are required for the value to be displayed, the remaining positions to the right will be filled with zeros. If there are more significant decimal digits in the value than in the mask, the extra digits will be truncated (not rounded):

Statement	Display
10: USING "####.##"	
20: PRINT 25	| 25.00|
30: PRINT −350.5	|−350.50|
40: PRINT 2.547	| 2.54|

Specifying Scientific Notation

A "^" character may be included in the mask to indicate that the number is to be displayed in scientific notation. The '#' and '.' characters are used in the mask to specify the format of the "characteristic" portion of the number, i.e., the part which is displayed to the left of the E. Two '#' characters should always be used to the left of the decimal point to provide for the sign character and one integer digit. The decimal point may be included, but is not required. Up to 9 '#' characters may appear to the right of the decimal point. Following the characteristic portion, the exponentiation character, E, will be displayed followed by one position for the sign and two positions for the exponent. Thus, the smallest scientific notation field would be provided by a mask of "##^" which would print numbers of the form '2E 99'. The largest scientific notation field would be "##.##########^" which would print numbers such as '−1.234567890E−12':

Statement	Display
10: USING "###.##^"	
20: PRINT 2	| 2.00E 00|
30: PRINT −365.278	|−3.65E 02|

Specifying Alphanumeric Masks

String constants and variables are displayed using the '&' character. Each '&' indicates one character in the field to be displayed. The string will be positioned at the left end of this field. If the string is shorter than the field, the remaining spaces to the right will be filled with spaces. If the string is longer than the field, the string will be truncated to the length of the field:

Statement	Display
10: USING "&&&&&&"	
20: PRINT "ABC"	|ABC |
30: PRINT "ABCDEFGHI"	|ABCDEF|

Mixed Masks

In most applications a USING mask will contain either all numeric or all string formatting characters. Both may be included in one USING mask, however, for certain purposes. In such cases, each switch from numeric to string formatting characters or vice versa marks the boundary for a different value. Thus, a mask of "#####&&&&" is a specification for displaying two separate values—a numeric value which is allocated 5 positions and a string value which is allocated 4 positions:

Statement	Display
10: PRINT USING "###.##&&";25;"CR"	|**25.00CR**|
20: PRINT −5.789;"DB"	|**−5.78DB**|

Remember: Once specified, a USING format is used for all output which follows until cancelled or changed by another USING verb.

APPENDIX D
EXPRESSION EVALUATION AND OPERATOR PRIORITY

When the SHARP PC-1350 is given a complex expression, it evaluates the parts of the expression in a sequence which is determined by the priority of the individual parts of the expression. If you enter the expression:

100/5+45

as either a calculation or as a part of a program, the PC-1350 does not know whether you mean:

$$\frac{100}{5+45} = 2 \quad \text{or} \quad \frac{100}{5} + 45 = 65$$

Since the PC-1350 must have some way to decide between these options, it uses its rules of operator priority. Because division has a higher "priority" than addition (see below), it will choose to do the division first and then the addition, i.e., it will choose the second option and return a value of 65 for the expression.

Operator Priority

Operators on BASIC of the SHARP PC-1350 are evaluated with the following priorities from highest to lowest:

Level	Operations
1.	Parentheses
2.	Variables and Pseudovariables
3.	Functions
4.	Exponentiation (^)
5.	Unary minus, negative sign (−)
6.	Multiplication and division (*, /)
7.	Addition and subtraction (+, −)
8.	Relational operators (<, <=, =, <>, >=, >)
9.	Logical operators (AND,OR,NOT)

When there are two or more operators at the same priority level the expression will be evaluated from left to right. (The exponentiation will be evaluated from right to left). Note that with A+B−C, for example, the answer is the same whether the addition or the subtraction is done first.

When an expression contains multiple nested parentheses, the innermost set is evaluated first and evaluation then proceeds outward.

For level 3 and 4, the last entry has a higher priority.
For example: $-2\,\hat{}\,4 \rightarrow -(2^4)$
$3\,\hat{}\,-2 \rightarrow 3^{-2}$

Sample Evaluation

Starting with the expression:

((3+5−2)*6+2)/10 ^ LOG 100

The PC-1350 would first evaluate the innermost set of parentheses. Since '+' and '−' are at the same level it would move from left to right and would do the addition first:

((8−2)*6+2)/10^LOG 100

Then it would do subtraction:

((6)*6+2)/10^LOG 100

or:

(6*6+2)/10^LOG 100

In the next set of parentheses it would do the multiplication first:

(36+2)/10^LOG 100

And then the addition:

(38)/10^LOG 100

or:

38/10^LOG 100

Now that the parentheses are cleared, the LOG function has the highest priority so it is done next:

38/10 ^ 2

The exponentiation is done next:

38/100

And last of all the division is performed:

0.38

This is the value of the expression.

APPENDIX E
KEY FUNCTIONS IN BASIC

ON BRK (ON)
Use to turn the PC-1350 power on when the auto power off function is in effect.
(BREAK)
- Depressing this key during program execution functions as a BREAK key and causes the program to interrupt execution.
- When pushed during manual execution, input/output command such as BEEP, CLOAD, etc., execution of the command is interrupted.

SHIFT
- The yellow key marked "SHIFT" must be used to designate a second function. (the material appearing in brown above each key.)
  Ex. SHIFT U → ? is entered.
- When storing something in or recalling something from the ReSerVe mode, press before the labelled key.

CLS
- Use to clear the contents of the entry and the display.
- Use to reset the error.

SHIFT CA
- Not only clears the display contents, but resets the computer to its initial state.
  —Initial state—
  - Resets the WAIT timer.
  - Resets the display format. (USING format)
  - Resets the TRON state (TROFF).
  - Resets the PRINT=LPRINT.
  - Resets error.

MODE
- Use to change the operational mode selection from RUN to PROgram or from PROgram to RUN.

SHIFT MODE
- Use to set the reserve mode.

0 ~ 9
- Numeric keys.

.
- Decimal point.
- Use to enter an abbreviation of a command/verb/function.
- Use to designate the decimal portion in USING format designation.

E
- Use to designate an exponent in scientific notation. (This key is a letter E key: upper case)

/
- Division key.

*****
- Multiplication key.

- Use to designate an array variable in the INPUT#, the PRINT#, etc.

[+]
- Addition key.

[−]
- Subtraction key.

[SHIFT] [^]
- Use for power calculation instructions.
- Use to specify the exponent display system for numerical data in USING statement instructions.

[SHIFT] [<]
[SHIFT] [>]
- Use when entering logical operators in IF sentence.

[DEF]
- When any one of eighteen keys (A, S, D, F, G, H, J, K, L, =, Z, X, C, V, B, N, M, SPaCe) is pushed after the depression of the [DEF] key, it starts to execute the program from the program line that has the same label as the key code depressed.

[A] ~ [Z]
- Alphabet keys. You are probably familiar with these keys from the standard typewriter keyboard.
  If pressed as is, an upper case character is displayed. If an alphabet key is pressed after pressing [SML], a lower case character is displayed.

[SPC]
- Use to provide space when entering programs or characters.

[=]
- In assignment statements, use to assign the contents (number or character) on the right for the variable specified on the left.
- Use when entering logical operators in IF sentence.

[SHIFT] [!]
["]
[#]
[$]
[%]
[&]
[SHIFT] [@]
- Use to designate these symbols.
- " : • Use to designate and cancel characters.
  • Use to specify labels.
- # : • Use with USING statement, to provide the instruction to define the display format of numerical data.
- $: • Use when assigning character variables.
- &: • Use with USING statement, to provide the instruction to define the display format of character string.
  • Use to designate hexadecimal number.
- @: • Used for reserve contents when the reserve key is used as a program key.
  Example: GOTO 100 @
- !%: • Used as a character string within " ".

[SHIFT] [?]
- Use to enter CLOAD?

[:]
- Use to divide two or more statements in one line.

[,]
- Use to provide pause between two equations, and between variables or comments.

[;]
- Use to provide multi-display (two or more values/contents displayed at a time).
- Use to provide pause between the instruction and the variable.

[(]
[)]
- Use to enter parentheses.

[►]
- Shifts the cursor to the right (press once to advance one position, hold down for automatic advance)
- Executes playback instructions.
- Call the cursor if not displayed while the contents are displayed.
- Clears an error condition in manual operation.

[◄]
- Shifts the cursor to the left (press once to advance one position, hold down for automatic advance)
- Otherwise the same as the [►] key.

[INS]
- Inserts one space (⊐ appears) of 1-step capacity between the address (N) indicated by the cursor and the preceding address (N-1).

[DEL]
- Deletes the contents of the address indicated by the cursor.

[SHIFT] [π]
- Use to designate pi (π).

[SHIFT] [√]
- Use to designate square root.

[ENTER]
- Enters a program line into the computer.
- Use when writing in programs
- Requests manual calculation or direct execution of a command statement by the computer.
- Used to restart a program temporarily stopped by an INPUT or PRINT command.

[SHIFT] [P↔NP]
- Use to set print and non-print mode when an optional printer is connected with the PC-1350.

[SML]
- Specifies and releases the lower case mode. (Turns on and off the SML symbol display.)
- The SML symbol is displayed when [SML] is pressed. Now, if [A] [B] and [C] are pressed, a b c is displayed. If [SML] is pressed again, the SML symbol disappears and upper case characters are entered.

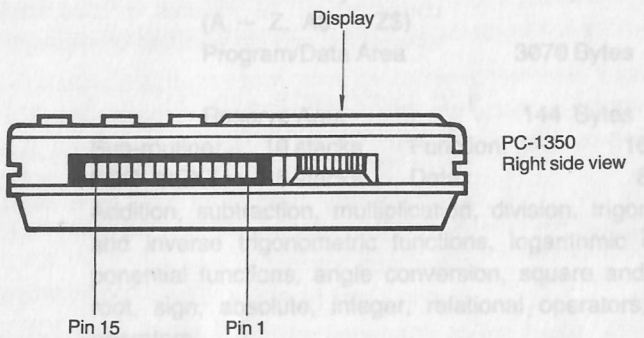
The [↑] and [↓] keys have the following functions, depending on designated modes, as well as the state of the computer.

Mode	State	[↓]	[↑]
RUN	Program being executed		
	Program is temporarily interrupted	To execute the next line	To display program line being executed or already executed, hold this key down.
	PRINT, LINE, PRESET, PSET, GPRINT statement being executed		
	PRINT statement just now executed		
	Under break		
	INPUT statement being executed	non function	
	Error condition during executing program		To display error-producing line, hold this key down.
	TRON condition	To execute debugging operation	To display program line being executed or already executed, hold this key down.
PRO	(When the mode is changed from RUN to PRO and program line is not being displayed)		
	Program is temporarily interrupted	To display the line interrupted	Same as left.
	Error condition	To display the line with error	Same as left
	Other condition	To display the first line	To display the last line
	(When the program line is being display)		
		To display the next program line	To display the preceeding program line
RESERVE			

- On the display, the [ENTER] key is the same as a space.
- If no key is entered in the key input request mode for approximately 11 minutes, the power is automatically turned off (auto-power off function).

APPENDIX F
SIGNALS USED IN THE SERIAL I/O TERMINAL

The PC1350 is equipped with a 15-pin connector for the serial I/O terminal. The pins used and their signals are described below



Pin Connecitons Used

Pin	Name	Symbol	I/O	Function
1	Frame Groud	FG		Ground for maintenance
2	Transmit Data	SD	o	DC output signal
3	Receive Data	RD	i	DC input signal
4	Request Data	RS	o	ON: sends carrier
5	Clear To Send	CS	i	ON: transmission enabled
7	Signal Ground	SG	—	Reference 0 voltage for all signals
8	Data Carrier Detect	CD	i	ON: carrier signal received
10		VC		Power supply
11	Recieve Ready	RR	o	ON: receive enabled
13		VC		Power supply
14	Data Terminal Ready	ER	o	ON: local terminal ready

Note: The voltage level of VC is the high level. The voltage level of SG is the low level.

Note: If a voltage which exceeds the allowable range (voltage level between SG and VC) of the PC-1350 is applied, the internal components may become damaged since the PC-1350 uses CMOS components.

APPENDIX G SPECIFICATIONS

Model:	PC-1350 Pocket Computer		
Processor:	8 bit CMOS CPU		
Programming Language:	BASIC		
System ROM:			40 K Bytes
Memory Capacity:	RAM:		
	System		1.6 K Bytes
	User		
	Fixed Memory Area (A ~ Z, A$ ~ Z$)		208 Bytes
	Program/Data Area		3070 Bytes
	Reserve Area		144 Bytes
Stack:	Sub-routine: 10 stacks	Function:	16 stacks
	FOR–NEXT: 5 stacks	Data:	8 stacks
Operators:	Addition, subtraction, multiplication, division, trigonometric and inverse trigonometric functions, logarithmic and exponential functions, angle conversion, square and square root, sign, absolute, integer, relational operators, logical operators.		
Numeric Precision:	10 digits (mantissa) + 2 digits (exponent).		
Editing Features:	Cursor left and right, line up and down, character insert, character delete.		
Memory Protection:	CMOS Battery backup.		
Serial Input/Output Features:			
	Standards:	Start-stop transmission (asynchronous) system. Only half duplex.	
	Baud Rates:	300, 600, 1200 Baud	
	Data Bits:	7 or 8 Bits	
	Parity Bits:	Even, odd, or no-parity	
	Stop Bit:	1 or 2 Bits	
	Connectors Used:	15-pin connector (for external equipment)	
	Output Signal Level:	C-MOS level (4–6 Volts)	
	Interfacing Signals:	Inputs:RS, CS, CD Outputs: SD, RS, RR, ER Others: SG, FG, VC	

Display: 4-line 24-digit liquid crystal display with 5×7 dot characters or 150×32 dot graphics.

Keys: 62 keys: Alphabetic, numeric, special symbols, and functions. Numeric pad. User defined keys.

Power Supply: 6.0V DC: Lithium cells.
Type: CR-2032×2

Power Consumption: 6.0V DC @ 0.03W
Approx. 250 hours when 555555555555555555555555. is on continuous display at an operating temperature of 20°C (68°F), this time may vary slightly with the operation method, etc.

- One hour operation per day allows the battery to be used for approx. 2.5 months. This is true for one hour operation consisting of 10 minutes of calculations or program executions and 50 minutes of displays.

Operating Temperature: 0°C – 40°C (32°F – 104°F)

Dimensions: 182(W)×72(D)×16(H) mm.
7-5/32″ (W)×2-27/32″ (D)×5/8″ (H)

Weight: Approximately 190g (0.42 lbs.) (with cells)

Accessories: Hard cover, two lithium cells (built-in), two keyboard templates and instruction manual.

Options: Plug-in Card type 8 K Bytes RAM (CE-201M),
16 K Bytes RAM (CE-202M)
Cassette Tape Recorder (CE-152)
Printer/Cassette Interface (CE-126P) etc.

APPENDIX H
USING PROGRAM DEVELOPED FOR THE PC-1210 SERIES, PC-1245 SERIES, PC-1260 SERIES OR PC-1401

Note: PC-1210 Series: PC-1210, PC-1211
PC-1245 Series: PC-1245, PC-1250, PC-1251
PC-1260 Series: PC-1260, PC-1261

The display units in the PC-1245 Series, PC-1210 Series, PC-1260 Series and PC-1401 Pocket Computers consist of either 1 or 2 lines. As a result, to change the contents shown on the display unit, the previous contents must first be cleared. In the PC-1350, however, the displayed contents (4 lines) scroll up with each new line.

Therefore, when executing programs which move characters or symbols, or programs using the CURSOR command in the PC-1260 Series, unexpected movements or displays may result. It is thus necessary to modify the program, such as by clearing the screen each time using the CLS command.

Program tapes (for the PC-1245 series or PC-1210 series) recorded with a number of programs loaded with the MERGE command cannot be used for the PC-1350. Merge the individual programs on the PC-1350.

Further, since the functions may differ slightly depending on the series of the pocket computer, the modifications to the program described below are required.

Modifications Required to PC-1245 series (PC-1245, PC-1250, PC-1251) Programs

When using the programs developed for the PC-1245 series, on the PC-1350, it is necessary to modify the following:

1. Multiplication without using the operator "*".
   On the PC-1245 series, the operator (*) for multiplication may be omitted, such as AB for A*B or CD for C*D. On the PC-1350, the multiplication operator (*) cannot be omitted since the computer treats two consecutive characters, such as AB or CD, as simple variables. Use the specification on the right hand side of the following example:

   (e.g.) A=SIN BC → A=SIN (B*C)

2. Definition of subscripted variables (such as A( )) by using the DIM statement:
On the PC-1245 series, if, for example, DIM A(30) is executed, memory locations for A(27) through A(30) are set aside as an extension of a fixed variable definition area. On the PC-1350, however, the execution of DIM A(30) reserves a separate memory area for array variables A(0) through A(30) for the array named A. When defining subscripted variables (such as A( )) as an extension of fixed variables, use the specification on the right hand side of the following example:

DIM A(30) → A(30)=0

3. Data I/O statement for tape files:
On the PC-1245 series, the execution of, for instance, the PRINT# C statement saves the contents of the variable C and all the subsequent variables to a tape file. On the PC-1350, however, the execution of the same statement saves the contents of the variable C only. To save the contents of a specific variable and all the subsequent variables, use the specification on the right hand side of the following examples:

(e.g.) PRINT#A → PRINT#A*
PRINT#C → INPUT#C*

Note: In this machine you cannot execute the programs for the PC-1250/1251 in which the POKE or CALL command is used. Executing such programs in this machine may cause the abnormal situation (for example, you can get no response from any key).

4. Value of a loop variable after completion of a FOR-NEXT loop:
The value of a loop variable obtained after the execution of a FOR-NEXT loop completed on the PC-1350 is different from that obtained on the PC-1245 series. If the value of a loop variable is used in a conditional expression in a PC-1245 series program, increment it by one when it is used on the PC-1350.

(e.g.) 10 FOR I=0 TO 10
⋮
50 NEXT I
60 IF I=10 THEN 100
Modify the value of I in line 60 as follows:
60 IF I=11 THEN 100
(On the PC-1350, the value of a loop variable must be incremented by one step value. The number of loop execution cycles remains the same, however.)

5. Exponent symbol "↤":
The PC-1350 uses the uppercase letter "E" for its exponent symbol. The following change are required:

A=1.234 ↤ 5 → A=1.234E5
B=↤ 6 → B =1E6

If a PC-1245 program is read from a tape file into the PC-1350, the change for the exponent symbol described just above will automatically be done by the PC-1350.

6. The character code of the PC-1245 series are partially different from those of the PC-1350.
When the following codes are designated by the CHR$ function, change the codes.

Character Code	PC-1245	PC-1350
39 (&27)	‾	,
91 (&5B)	√	[
92 (&5C)	¥	\
93 (&5D)	π	]
96 (&60)	↤	‾
250 (&EA)	—(Error)	π
251 (&FB)	—(Error)	√
252 (&FC)	—(Error)	

Note: As shown above, the PC-1350 does not have the characters ↤.

Additional modifications

1. The PC-1245, PC-1250, and PC-1251 use a line number ranging from 1—999, whereas this model, as well as the PC-1260/1261, has an extended line number ranging from 1—65279. Therefore, the line number uses 3 bytes in RAM (PC-1245 series uses 2 bytes). The modification is carried out automatically when the program is loaded through the cassette tape. However, there is a possibility of memory overflow (ERROR 6) when loading or executing a long program. Further, when a single line is close to 80 bytes long, this modification may result in the clearing of the end of the line.

2. In loading a program of the PC-1245 series through the cassette tape, the computer will remain BUSY for one to two seconds after the tape has stopped due to modification of the line number (2 bytes to 3 bytes) as mentioned previously. During this period, symbol "*" will be displayed at the right bottom column of the display as in loading a program.

Note: The PC-1245 series cannot read from a tape which contains programs developed for the PC-1350.

Modifications Required to PC-1401

(1) Modifications to the defined [,] key
The [,] key is not a defined key in this computer. Therefore, for the programs which define the [,] key, define another key.
Example:
100 ",": → 100 "=":

(2) Other notes

The PC-1401 contains more function commands than the PC-1350. Therefore, an error will occur during execution if a program which uses commands not available in the PC-1350, is written or read from a tape. If a program is read from a tape and contains commands not available in the PC-1350, those commands will be substituted with "~" and then displayed.

Modifications Required to PC-1210 Series

To use PC-1210 Series programs on the PC-1350, they must be modified in the same way as PC-1245 Series programs (except items 2 and 6). In addition, the following modifications are necessary.

(1) IF Statement

If, for example,

```
50 IF A>L PRINT "A" (display "A" if A>L)
```

is found in the program for the PC-1210 Series Pocket Computers, it is interpreted as

```
50 IF A>LPRINT "A" (Print out "A" if A>)
```

and results in an error when it is entered though the keyboard.

The error occurs because a command which does not exist in the PC-1210 Series does in fact exist in the PC-1350.

To solve this problem, insert a THEN command into the IF statement as follows.

```
5Ø IF A>L THEN PRINT "A"
```

(2) Specified Format in USING

The function of the USING command differs between the PC-1350 and the PC-1210 Series as follow.

[Example]

```
1Ø A = −123.456
2Ø PAUSE USING "####.##"; A
3Ø PAUSE A, USING "####"; A
```

Executing this program displays the following.

```
* PC-1210/PC-1211          −123.45
                   −123      −123
* PC-1350                  −123.45
                −123.45      −123
```

For the execution of line 3Ø in the PC-1210 Series, the display on the left side also follows the displayed format on the right side. In the PC-1350, the display follows the previous specified format. This applies not only to the PAUSE command, but also to the PRINT and LPRINT commands.

(3) Omitting ")"

In the PC-1210 Series, the ")" which comes immediately before the [ENTER] or : (colon) can be omitted. It cannot be omitted in the PC-1350.

Therefore, be sure to add the ")" to the program if omitted.

(4) Print Command

The PC-1350 has a PRINT command for displays and a LPRINT command for printing. However, all PRINT commands can be used for printing if PRINT = LPRINT is specified (See page 140.)

The PC-1210 Series does not have the LPRINT command. To print using a PC-1210 Series program, add PRINT = LPRINT to the program or, execute manually.

(5) Variables

When the RUN command has been executed in the PC-1210 Series, all variables are retained. In the PC-1350, however, all variables from A(27) and on are cleared. (See page 45).

Therefore, if there is a need to retain variables at the start of program execution, start the program execution using the GOTO command or function defined keys.

Modifying Required to PC-1260 Series

Modifying the Character Code

In the PC-1260 Series, the character for character code 96 (&6Ø) is a space, whereas in the PC-1350 it is a ' (single quote).

Therefore, if a space has been specified using code 96 in the CHR$ command, change it to code 32 ($2Ø)

Program Examples

Probably you have acquired knowledge on a number of program commands as you have proceeded to the function description up to this page. It is necessary,however, to generate actual programs by yourself in addition to reading the instruction manual, so that you can generate programs freely using BASIC language. Like driving a car or playing tennis that can be improved by actual practice, you can improve your programming only by generating as many programs as possible regardless of your skill. It is also important for you to refer to programs generated by others. For your reference, the following pages contain a variety of programs using BASIC commands.

(Sharp Corporation and/or its subsidiaries assume no responsibilities or obligations to any losses or damages that could arise through the use of the software programs employed in this instruction manual.)

CONTENTS

(program title) (page)

- SOFTLANDING GAME 231
- NUMBER OF DAYS REVERSE NUMBER OF DAYS CALCULATION 235
- ABC ANALYSIS 240
- MATRIX 246
- MODIFIED MOVING AVERAGE 251
- SINGLE ARRANGEMENT METHOD 256
- F TEST 262

* The display area of the PC-1350 displays 4 lines, but with key operation sequence in partial program, only the parts that are actually displayed are recorded.

* The number of bytes used in each program is recorded at the end of each program.

Program Title: SOFTLANDING GAME

■ OVERVIEW

Nearly out of fuel ! Emergency landing !

How do you land a rocket with little fuel remaining? The rocket is in free fall. The rocket engine is fired to slow down the free falling rocket. If ignition takes place too soon or too much feul is used, then the rocket becomes dust aroud the planet. Or if you are too late, then the rocket crashes into surface of the planet.
Well, try out your skill and courage and try to safely land this rocket. The secret of succeeding in your mission is to control the ignition time and the fuel cosumption.

■ INSTRUCTIONS

1. Starting program with DEF A keys pressed.
2. Input the amount of fuel used by keys 0 to 9. When landing the rocket, land by skilfully regulating the fuel. When you have succeeded, the remaining fuel will be displayed and if you fail, "Goodbye!!" will be displayed. The program will not continue unless you input whether you will play again or not.

■ CONTENTS

Gravity is set to be 5m/ (unit time)^2.
If 5 units of fuel per unit of time are burnt, then gravity is offset.

$$H = H_0 + V_0 t + \frac{1}{2} a t^2$$

$$V = V_0 + at$$

$$V^2 = V_0^2 + 2aH$$

$$H_0 = 5\emptyset\emptyset,\ V_0 = -5\emptyset,\ F_0 = 2\emptyset\emptyset$$

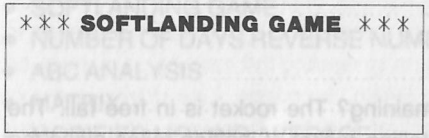
H : height
V : speed
a : gravitational acceleration
t : time

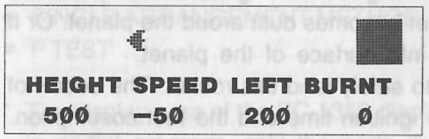
H_0 : initial height
V_0 : initial speed
F_0 : initial fuel
F : fuel burned

Note: You can make the game more interesting by changing the wait time numerical values in line 4Ø, the initial fuel in line 5Ø and the initial height in line 6Ø.

■ KEY OPERATION SEQUENCE

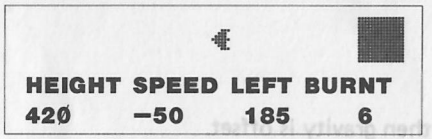
1) DEF A [Starting program]





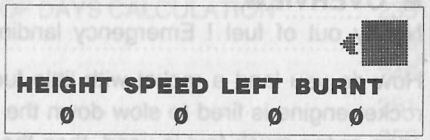
2) 9 [Input fuel burnt]

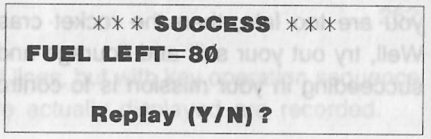
3) 6



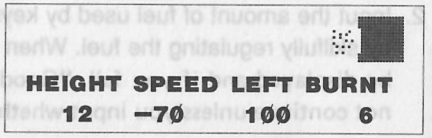
Input the burnt fuel in the same way and carry on with the game.

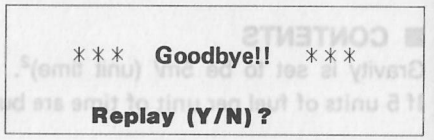
[Display examples for when you have succeeded]





[Display examples for when you have failed]

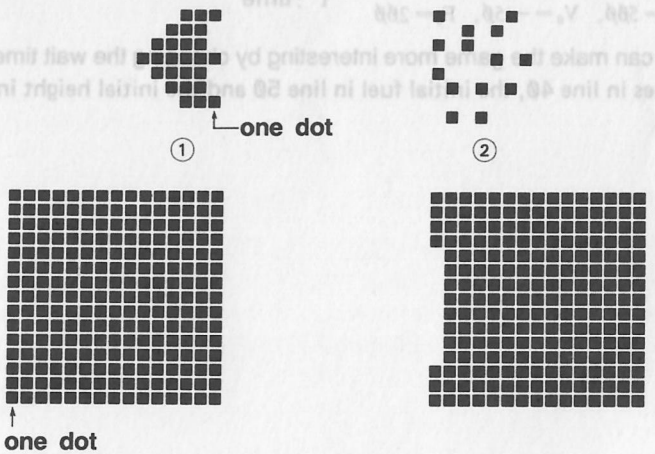




* Input Y if you want to play again.

Reference

Dot patterns for the Softlanding Game Display.



■ PROGRAM LIST

```
10:"A": WAIT 0: CLEAR :
   USING : CLS
20:S=-50:A=0:D$="":V=30
30:CURSOR 0,1: PRINT "*
   ** SOFTLANDING GAME
   ***": BEEP 3: CLS
40:DATA "TIME=",5
50:DATA "FUEL=",200
60:DATA "HEIGHT=",500
70:RESTORE : READ B$,W,
   B$,F,B$,H
80:WAIT W
90:FOR P=1 TO 2:
   GCURSOR (135,P*7):
   GPRINT "FFFFFFFFFFFF
   FFFFFFFFFFFFFFFFFF":
   NEXT P
100:CURSOR 0,2: PRINT "H
   EIGHT SPEED LEFT BUR
   NT"
110:GOSUB 300
120:L=V
130:IF F<=0 LET C=0:
   BEEP 1: GOTO 180
140:BEEP 1:D$= INKEY$
150:IF D$="" LET C=A:
   GOTO 180
160:C= VAL D$
170:A=C
180:IF C>F LET C=F
190:F=F-C:X=C-5:H=H+S+X/
   2:S=S+X
200:IF H>0 LET V=(500-H)
   /5+30: GOTO 110
210:IF ABS H<5 OR ABS S<
   5 LET H=0:S=0:L=V:V=
   129: GOSUB 300:
   BEEP 5: CLS : PRINT
   "   *** SUCCESS  **
   *": GOTO 250
220:GCURSOR (L+11):
   GPRINT "000000000000
   "
230:GCURSOR (129,11):
   GPRINT "15A006A00A21
   08"
240:BEEP 3: CLS :
   CURSOR 0,1: PRINT "
    *** Goodbye!!  ***
   ": GOTO 260
250:PRINT USING "####";"
      FUEL LEFT =";F
260:CURSOR 0,3: PRINT "
      Replay (Y/N) ?"
270:Z$= INKEY$ : IF Z$="
   Y" THEN 10
280:IF Z$="N" END
290:GOTO 270
300:CURSOR 0,3: PRINT
   USING "#####";H;
   USING "######";S;F;
   USING "#####";C
310:GCURSOR (L,11):
   GPRINT "000000000000
   "
320:GCURSOR (V,11):
   GPRINT "081C3E7F7F49
   "
330:RETURN
```

779 bytes

■ MEMORY CONTENTS

A	√
B$	√
C	fuel burnt
D$	fuel burnt
E	
F	initial fuel fuel left
G	
H	initial height height
I	
J	
K	
L	√
M	
N	
O	
P	loop counter
Q	
R	
S	speed
T	
U	
V	√
W	wait time
X	√
Y	
Z$	√

Program Title: NUMBER OF DAYS · REVERSE NUMBER OF DAYS CALCULATION

■ OVERVIEW

With this program two types of day calculation are possible. When you want to know how many days have passed from one date to the other, or when you want to know what day, month and year it will be at a number of days later from a certain date, you can soon know the results if you use this program. Just enter the start date and the target date, or the start date and the number of days. Try calculating for example, the number of days from your birthday up to today or what day, month and year it will be in 10,000 days from now.

■ INSTRUCTIONS

[DEF] [A] operation ..... Number of days calculation

1. When you input each start year, month and day according to the dsplay, operate [N] [ENTER] for to amend the details and [Y] [ENTER] for when you do not wish to amend them, to the OK? prompt.
2. Then if you input the target date together with the start date the number of days will be displayed.
   Furthermore, once the number of days has been displayed it is possible to once again input from the start date by operating [ENTER] .
3. When you end the program, control is ended by operating [ENTER] when it is waiting for the start year input.

[DEF] [B] operation ..... Reverse number of days calculation

Operate as mentioned above.

Note: Number of days calculated by this program doesn't include the base day. If you want the number that includes the base day, please change the program as follows:

```
260:CURSOR 0,3: PRINT "N
    O. OF DAYS =>";
    USING "######";X+1
```

■ EXAMPLES

1. Number of days calculation
   The number of days from 7th of February 1961 to 9th of May 1984.
2. Reverse number of days calculation
   The day, month and year 8492 days from 7th February 1961.

■ KEY OPERATION SEQUENCE

[In the case of example 1]

1) [DEF] [A] [Start program]

```
** NO. OF DAYS CALC. **
START  Y=?      M=   D=
TARGET Y=       M=   D=
```

2) 1961 [ENTER] 2 [ENTER] 7 [ENTER]

```
** NO. OF DAYS CALC. **
START  Y=1961   M=2  D=7
TARGET Y=       M=   D=
                OK(Y/N)?_
```

3) [Y] [ENTER]

```
** NO. OF DAYS CALC. **
START  Y=1961   M=2  D=7
TARGET Y=?      M=   D=
```

4) 1984 [ENTER] 5 [ENTER] 9 [ENTER]

```
** NO. OF DAYS CALC. **
START  Y=1961   M=2  D=7
TARGET Y=1984   M=5  D=9
                OK(Y/N)?_
```

5) [Y] [ENTER] [Display date]

```
** NO. OF DAYS CALC. **
START  Y=1961   M=2  D=7
TARGET Y=1984   M=5  D=9
PLEASE WAIT !!
```

```
** NO. OF DAYS CALC. **
START  Y=1961   M=2  D=7
TARGET Y=1984   M=5  D=9
NO. OF  DAYS =>8492_
```

6) [ENTER] [Waiting for start year input]

```
** NO. OF DAYS CALC. **
START  Y=?      M=   D=
TARGET Y=       M=   D=
```

7) [ENTER] [Program end]

```
>
```


■ KEY OPERATION SEQUENCE

[In the case of example 2]

1) [DEF] [B] [Start program]

```
** NO. OF DAYS CALC. **
START  Y=?      M=   D=
DAYS    =
```

2) 1960 [ENTER] 2 [ENTER] 7 [ENTER]

```
** NO. OF DAYS CALC. **
START  Y=1960   M=2  D=7
       DAYS=
                OK(Y/N)?_
```

3) [N] [ENTER] [Input error has been found]

```
** NO. OF DAYS CALC. **
START  Y=?      M=   D=
DAYS    =
```

4) 1961 [ENTER] [Correct data input]

```
** NO. OF DAYS CALC. **
START  Y=1961   M=?  D=
DAYS    =
```

.
.
input as above
.
.

5) [Y] [ENTER]

```
** NO. OF DAYS CALC. **
START  Y=1961   M=2  D=7
DAYS    =?
```

6) 8492 [ENTER] [Number of days input]

```
** NO. OF DAYS CALC. **
START  Y=1961   M=2  D=7
DAYS        =8492
                OK(Y/N)?_
```

7) [Y] [ENTER] [Target date display]

```
** NO. OF DAYS CALC. **
START  Y=1961   M=2  D=7
DAYS        =8492
PLEASE WAIT!!
```

```
** NO. OF DAYS CALC. **
START  Y=1961   M=2  D=7
DAYS        =8492
TARGET Y=1984.  M=5. D=9._
```

8) [ENTER] [Waiting for start year input]

```
** NO. OF DAYS CALC. **
START  Y=?      M=   D=
DAYS    =
```

9) [ENTER] [Program end]

```
>
```

■ PROGRAM LIST

```
10:REM ****************
20:REM NO. OF DAYS CALC
30:REM ****************
40:"A": CLEAR : CLS :
   WAIT 0: USING :A=0:
   GOTO 50
45:"B": CLEAR : CLS :
   WAIT 0: USING :A=1
50:PRINT "** NO. OF DAY
   S CALC. **"
60:BEEP 1
80:CURSOR 0,1: PRINT "S
   TART  Y=";"      M=";
   "   D=   "
85:IF A=1 CURSOR 0,2:
   PRINT "DAYS    =
          ": GOTO 95
90:CURSOR 0,2: PRINT "T
   ARGET Y=";"      M=";
   "   D=   "
95:CURSOR 0,3: PRINT "
    "
97:IF C=1 THEN 180
100:CURSOR 9,1: INPUT R:
    GOTO 120
110:CLS : END
120:CURSOR 16,1: INPUT S
130:CURSOR 21,1: INPUT T
140:CURSOR 0,3: PRINT "

     "
150:CURSOR 13,3: INPUT "
    OK(Y/N)?";F$
160:IF F$="N" THEN 80
165:IF F$<>"Y" GOTO 150
170:CURSOR 0,3: PRINT "

     "
180:CURSOR 9,2: INPUT U
189:IF A=1 LET X=U:
    GOTO 210
190:CURSOR 16,2: INPUT V
200:CURSOR 21,2: INPUT W
210:CURSOR 13,3: INPUT "
    OK(Y/N)?";F$
220:IF F$="N" LET C=1:
    GOTO 85
225:IF F$<>"Y" THEN 210
230:CURSOR 0,3: PRINT "
    PLEASE WAIT !!
     "
235:IF A=1 THEN 400
240:H=R:G=S:I=T: GOSUB 3
    00:J=I:H=U:G=V:I=W:
    GOSUB 300:X=I-J
250:BEEP 1: CURSOR 0,3:
    PRINT "
                "
260:CURSOR 0,3: PRINT "N
    O. OF DAYS =>";
    USING "######";X
270:CURSOR 20,3: INPUT "
     ";K
280:USING :C=0: GOTO 80
300:IF G-3>=0 LET G=G+1:
    GOTO 320
310:G=G+13:H=H-1
320:I= INT (365.25*H)+
    INT (30.6*G)+I:I=I-
    INT (H/100)+ INT (H/
    400)-428: RETURN
400:H=R:G=S:I=T: GOSUB 6
    00:J=Q+X:G=1:I=0
410:H=H+1: GOSUB 600
420:IF J>Q THEN 410
430:H=H-1
440:G=G+1: GOSUB 600
450:IF J>Q THEN 440
460:G=G-1: GOSUB 600:I=J
    -Q:H=H-B: BEEP 1:
    CURSOR 0,3: PRINT "
                "
470:CURSOR 0,3: PRINT "T
    ARGET Y=";"      M=";
    "   D=   "
475:CURSOR 9,3: PRINT H:
    CURSOR 16,3: PRINT G
    : CURSOR 21,3:
    PRINT I
480:CURSOR 21,2: INPUT "
    ";K
490:GOTO 80
600:IF G-3>=0 LET Q=G+1:
    P=H: GOTO 620
610:Q=G+13:P=H-1
620:Q= INT (365.25*P)+
    INT (30.6*Q)+I:Q=Q-
    INT (P/100)+ INT (P/
    40)-428: RETURN
```

1244 bytes

■ MEMORY CONTENTS

A	√
B	√
C	√
D	
E	
F$	√
G	√
H	√
I	√
J	√
K	√
L	
M	
N	
O	√
P	√
Q	√
R	start year
S	start month
T	start day
U	target year
V	target month
W	target day
X	number of days
Y	
Z	

Program Title: ABC ANALYSIS

Thermal printer cassette interface CE-126P required

■ OVERVIEW

This program carries out product ranking. It is important for sales supervisors to know what position each product occupies in the general sales returns. When you apply ABC analysis which is essentially a means of stock control and input the amount sold and the product number this program will compute the sales rate and the sumtotal rate and print this out for you as well as place them in ABC ranking.

■ INSTRUCTIONS

1. Program starts at [R] [U] [N] [ENTER].
   Input the product number and the sales returns according to the display.
   When it is waiting for the product number to be input, you can shift to control 2 if you operate [ENTER] only.
2. Select whether you will amend the data you input next.
   When selecting data to amend, operate [Y] [ENTER] and if the data is wrong input the correct data, as the sales returns and the product number are displayed by operating [ENTER] only.
   When there is no need to amend, operate [N] [ENTER].
3. Output the sales rate, the sumtotal rate and ABC rank placing.
   When you output the results to printer, operate [P] [ENTER].
   When you display the results, operate [L] [ENTER]. Furthermore, when everything cannot be displayed input [U] (up) and [D] (down).
   Operate [E] [ENTER], the process is completed.
4. When you print out the results or input [Q] after it has been displayed, once again it will shift to control 3.

Note: Input for product numbers is possible for up to 7 letters and for sales returns for up to 6 digits at the most. Furthermore, the maximum of cases which can be entered is 26.

■ EXAMPLES

The sales rate and the sumtotal rates are calculated from the data below and ABC ranking is carried out.

Product No.	Sales returns
TR-300	89,000
BT-650	17,000
GZ-90	23,000
RS-15	65,000
JO-230	56,000
PQ-180	108,000
YC-30	21,000

■ PRINTED OUTPUTS

```
   ** ABC  ANALYSIS **

*PROD. NO.*          (RANK)
 SALES   %        SUM   %
-------------------------
* PQ-180  *           ( A )
108000 28.4   108000 28.4
-------------------------
* TR-300  *           ( A )
89000  23.4   197000 51.9
-------------------------
* RS-15   *           ( B )
65000  17.1   262000 69.1
-------------------------
* JO-230  *           ( B )
56000  14.7   318000 83.9
-------------------------
* GZ-90   *           ( B )
23000   6.0   341000 89.9
-------------------------
* YC-30   *           ( B )
21000   5.5   362000 95.5
-------------------------
* BT-650  *           ( C )
17000   4.4   379000 100
-------------------------
```

■ KEY OPERATION SEQUENCE

1) [R] [U] [N] [ENTER]

```
(1) PRODUCT NO.   SALES
    ?
```

2) TR-300 [ENTER] [1st product no. input]

```
(1) PRODUCT NO.   SALES
    TR-300        ?
```

3) 88000 [ENTER]
[1st product sales returns input]

```
(1) PRODUCT NO.   SALES
    TR-300        88000
(2) PRODUCT NO.   SALES
    ?
```

:

input as above

:

4) YC-30 [ENTER]
[7th product number input]

```
(6) PRODUCT NO.   SALES
    PQ-180        108000
(7) PRODUCT NO.   SALES
    YC-30         ?
```

5) 21000 [ENTER]
[7th product sales returns input]

```
(7) PRODUCT NO.   SALES
    YC-30         21000
(8) PRODUCT NO.   SALES
    ?
```

6) [ENTER] [select data confirmation?]

```
CONFIRM OR AMEND (Y/N)_
```

7) [Y] [ENTER] [confirm data]

```
(1) TR-300->  ?
```

8) [ENTER]

```
(1) TR-300->  ?
    88000 ->  ?
```

9) 89000 [ENTER] [correct data input]

```
(1) TR-300->  ?
    88000 ->  89000
(2) BT-650->  ?
```

.
.

carry out the same as above

.
.

10) [ENTER]

```
(6) PQ-180->  ?
    108000 ->  ?
(7) YC-30 ->  ?
    21000 ->  ?
```


■ KEY OPERATION SEQUENCE

11) [ENTER]
[select reconfirmation of data?]

```
CONFIRM OR AMEND (Y/N)_
```

12) [N] [ENTER]
[will not confirm data]

```
LCD or PRINTER or END
        (L/P/E)?_
```

13) [L] [ENTER]
[output method select display]

PR.NO.	RATE(%)	SUM(%)	R
PQ-180	28	28	A
TR-300	23	51	A
RS-15	17	69	B

14) [D]
[display confirmation by scroll down]

PR. NO.	RATE(%)	SUM(%)	R
TR-300	23	51	A
RS-15	17	69	B
JO-230	14	83	B

.
.

carry out the same as above

.
.

15) [D]

PR. NO.	RATE(%)	SUM(%)	R
GZ-90	6	89	B
YC-30	5	95	B
BT-650	4	100	C

16) [Q]
[select output method once again]

```
LCD or PRINTER or END
        (L/P/E)?_
```

17) [P] [ENTER]
[output method select print]

```
LCD or PRINTER or END
        (L/P/E)?_
```

18) [E] [ENTER]
[program compdetion]

```
>
```

■ PROGRAM LIST

```
40:CLEAR : CLS : WAIT 0
   :A=26:B=14: DIM B$(A
   )*B,C(A),D(A),K$(A)*
   1,Z$(0)*6
50:FOR I=1 TO A:
   CURSOR : IF I>2
   GOSUB 700: PRINT " "
60:CURSOR 0,(I>1)*2:
   PRINT "("; STR$ I;")
   PRODUCT NO.  SALES":
   PRINT " "
70:BEEP 1: CURSOR 2+
   LEN STR$ I,(I=1)+(I>
   1)*3: INPUT B$(I)
80:IF B$(I)="" LET C=I-
   1:I=A: GOTO 130
90:B$(I)= LEFT$ (B$(I),
   7)
95:CURSOR 2+ LEN STR$ I
   ,(I=1)+(I>1)*3:
   PRINT B$(I);"
       ";
100:CURSOR 14+ LEN STR$
   I,(I=1)+(I>1)*3:
   BEEP 1: INPUT Z$(0)
107:CURSOR 14+ LEN STR$
   I,(I=1)+(I>1)*3:
   PRINT Z$(0);"   ";
110:C(I)= VAL Z$(0): IF
   C(I)<=0 BEEP 1:
   GOTO 100
120:E=E+C(I):C=I:Z$(0)="
   "
130:NEXT I: IF C<2 BEEP
   2: GOTO 40
140:BEEP 1: CLS : INPUT
   "CONFIRM OR AMEND(Y/
   N)";Z$
150:IF Z$="N" GOSUB 520:
   GOTO 240
160:IF Z$<>"Y" THEN 140
170:CLS : FOR I=1 TO C
180:CURSOR : IF I>2
   GOSUB 700: PRINT " "
190:CURSOR 0,(I>1)*2:
   PRINT "("; STR$ I;")
   ";B$(I);" -> ":
   CURSOR 6+ LEN STR$ I
   + LEN B$(I),(I>1)*2
200:INPUT B$(I):B$(I)=
   LEFT$ (B$(I),7)
210:CURSOR 2+ LEN STR$ I
   ,(I=1)+(I>1)*3:
   PRINT STR$ C(I);" ->
    ": CURSOR LEN STR$
   C(I)+7,(I=1)+(I>1)*3
   :Z$(0)= STR$ C(I)
220:INPUT Z$(0): IF VAL
   Z$(0)<=0 BEEP 1:
   GOTO 210
230:E=E-C(I):C(I)= VAL Z
   $(0):E=E+C(I): NEXT
   I: CLS : GOTO 140
240:CLS :H=0:V=0: PRINT
   "LCD or PRINTER or E
   ND"
250:BEEP 1: CURSOR 7,1:
   INPUT "(L/P/E)?";Z$
260:IF Z$="E" BEEP 3:
   CLS : END
270:IF Z$="P" THEN 430
280:IF Z$<>"L" THEN 240
290:CLS : CURSOR 0,0:
   PRINT "PR. NO. RATE(
   %)": CURSOR 16,0:
   PRINT " SUM(%)R":Y=1
300:FOR I=1 TO 3-(C=2):
   CURSOR 0,I: GOSUB 70
   0:V=D(Y-1)
310:CURSOR 0,I: PRINT B$
   (Y): CURSOR 8,I:O$=
   STR$ ( INT (D(Y)-V))
315:IF LEN O$=1 LET O$="
    "+O$
317:PRINT O$
320:CURSOR (64+D(Y)/2)/6
   -.9,I: PRINT USING "
   ####";D(Y): CURSOR 2
   3,I: PRINT K$(Y)
330:LINE (66,I*8+1)-(66+
   (D(Y)-V)/2,I*8+5),BF
340:LINE (66+(D(Y)-V)/2,
   I*8+1)-(66+D(Y)/2,I*
   8+5),B
350:Y=Y+1: NEXT I
360:LINE (2,7)-(149,7):
   LINE (2,31)-(149,31)
   : LINE (2,0)-(2,31)
370:LINE (48,0)-(48,31):
   LINE (143,0)-(143,31
   ): LINE (149,0)-(149
   ,31)
380:IF INKEY$ ="" GOTO 3
   80
390:IF INKEY$ ="Q" GOTO
   240
400:IF INKEY$ ="D" AND Y
   <C+1 BEEP 1:Y=Y-2:
   GOTO 300
410:IF INKEY$ <>"U" OR Y
   <5 BEEP 2: GOTO 380
420:Y=Y-4: BEEP 1: GOTO
   300
430:LPRINT "  ** ABC  AN
   ALYSIS **": LPRINT
440:LPRINT "*PROD. NO.*
          (RANK)"
450:LPRINT " SALES  %
        SUM  %"
460:FOR I=1 TO C: GOSUB
   690:H=H+C(I)
470:LPRINT "* "; USING "
   &&&&&&&";B$(I);
   USING ;" *        (
   "K$(I);" )"
480:LPRINT USING "&&&&&&
   "; STR$ C(I); USING
   "###.#";D(I)-V;
   USING "#########";H;
490:IF D(I)<>100 USING "
   ###.#": GOTO 510
500:USING "####"
510:LPRINT D(I):V=D(I):
   NEXT I: GOSUB 690:
   LPRINT : LPRINT :
   GOTO 240
520:L= INT (C/2+1):R=C
530:IF L>1 LET L=L-1:X=C
   (L):B$(0)=B$(L):
   GOTO 560
540:X=C(R):B$(0)=B$(R):C
   (R)=C(1):B$(R)=B$(1)
   :R=R-1
550:IF R<=1 LET C(1)=X:B
   $(1)=B$(0): GOTO 630
560:J=L
570:I=J:J=2*J
580:IF J>R THEN 620
590:IF J<R THEN IF C(J)>
   C(J+1) LET J=J+1
600:IF X<=C(J) THEN 620
610:C(I)=C(J):B$(I)=B$(J
   ): GOTO 570
620:C(I)=X:B$(I)=B$(0):
   GOTO 530
630:FOR I=1 TO C
640:H=H+C(I):D(I)=H/E*10
   0
650:IF D(I)<=66 OR I=1
   LET K$(I)="A": GOTO
   680
660:IF D(I)<=99 LET K$(I
   )="B": GOTO 680
670:K$(I)="C"
680:NEXT I: RETURN
690:USING : LPRINT "----
   ----------
   ": RETURN
700:PRINT "
                ":
   RETURN
```

2096 bytes

■ MEMORY CONTENTS

A	maximum number of data cases
B	product number letter length
C	number of data cases
D	
E	general sales returns
F	
G	
H	√
I	loop counter
J	loop counter
K	
L	√
M	
N	
O$	√
P	
Q	
R	end pointer
S	
T	
U	
V	√
W	
X	√
Y	√
Z$	√
C(A)	sales returns
D(A)	√
B$(A)	name of client
K$(A)	√
Z$(0)	√

Program Title: MATRIX

Thermal printer
cassette interface
CE–126P required

■ OVERVIEW

This program helps you not to make simple mistakes.
In spite of the calculating method itself being easy to understand, it is time consuming through simple mistakes occurring at the stage for carrying out the calculations. This program is for that purpose. Let us introduce you here to a program that searches for those troublesome matrix values. Correction and print out of the input values is also possible.

■ INSTRUCTIONS

1. To start the program, depress DEF A keys.
   - Enter the degrees of the matrix, followed by each element residing in the matrix.
   - When all data has been entered, select whether or not to have the data recorded on the printer.
2. Next, amend any data if required.
   - When amending an element, enter the row and column number of the data element to be amended.
   - Once all data is correct, answer N to the AMEND DATA (Y/N) prompt.
3. Calculation Output
   - After data correction, the option is given to print the data again. If no further processing is required, answer N.

■ CONTENTS

This program converts the matrix into a triangular matrix by using the sweeping-out method, then gets the answer.
Assume that a matrix is $\{a_{ij}\}$ $(i, j = 1 \sim n)$

$$P = a_{mm} \qquad (m = 2 \sim n)$$

$$q = a_{im}/P \qquad (i = 1 \sim m - 1)$$

$$a_{ij} = a_{ij} - q \cdot a_{mj} \qquad (j = 1 \sim m)$$

On calculation, the following is obtained:

$$a_{ij} = 0 \text{ for } i<j$$

This results in: det $= a_{11} \cdot a_{22} \cdot a_{33} \cdot \cdots \cdot a_{nn}$
With P = 0, however, during computation, an error results, since it's impossible to calculate.

■ EXAMPLE

Search for the following determination point.

$$\begin{bmatrix} 4 & 7 & 1 & 8 \\ 5 & 3 & 9 & 3 \\ 2 & 5 & 3 & 5 \\ 1 & 4 & 7 & 8 \end{bmatrix}$$

■ PRINTED OUTPUTS

```
A(1,1)=4.
A(1,2)=7.
A(1,3)=1.
A(1,4)=8.
A(2,1)=5.
A(2,2)=3.
A(2,3)=9.
A(2,4)=3.
A(3,1)=2.
A(3,2)=5.
A(3,3)=3.
A(3,4)=5.
A(4,1)=1.
A(4,2)=4.
A(4,3)=7.
A(4,4)=8.

det= -525.
```

■ KEY OPERATION SEQUENCE

1) DEF A [Start program]

```
DEGREES=_
```

2) 4 ENTER [Degrees input]

```
DEGREES=4
A(1,1)=?
```

3) 4 ENTER [Element input]

```
DEGREES=4
A(1,1)=4
A(1,2)=?
```

4) 6 ENTER

```
DEGREES=4
A(1,1)=4
A(1,2)=6
A(1,3)=?
```

.
.
.

input as above

.
.
.

5) 8 ENTER

```
A(4,2)=4
A(4,3)=7
A(4,4)=8
PRINT OUT DATA (Y/N)_
```

6) N ENTER

```
A(4,3)=7
A(4,4)=8
PRINT OUT DATA (Y/N)N
AMEND DATA (Y/N)_
```

7) Y ENTER [Amended data]

```
X=?          Y=
```

8) 1 ENTER [Input no. of columns]

```
X=1          Y=?
```

9) 2 ENTER [Input no. of lows]

```
X=1          Y=2
A(1,2)=6.
A(1,2)=?
```

10) 7 ENTER [Correct data input]

```
X=1          Y=2
A(1,2)=6.
A(1,2)=7
AMEND DATA (Y/N)_
```

11) N ENTER

```
A(1,2)=6.
A(1,2)=7
AMEND DATA (Y/N) N
PRINT OUT DATA (Y/N)_
```

12) Y ENTER [Print out data list after amendment]

```
A(1,2)=7
AMEND DATA (Y/N)N
PRINT OUT DATA (Y/N)Y
AMEND DATA (Y/N)_
```

13) N ENTER [Output results complete program]

```
>
```


■ PROGRAM LIST

```
 10:REM ****************
 20:REM * MATRIX CALC. *
 30:REM ****************
 40:"A": CLEAR : CLS :
    WAIT 0
 50:INPUT "DEGREES=";N
 60:N=N-1
 70:DIM A(N,N)
 80:GOSUB 230
 90:F=0
100:INPUT "PRINT OUT DAT
    A (Y/N)";O$
110:IF O$="N" THEN 140
120:IF O$<>"Y" THEN 90
130:GOSUB 350
140:INPUT "AMEND DATA (Y
    /N)";O$
150:IF (O$="N")*(F=0)
    THEN 200
160:IF (O$="N")*(F=1)
    THEN 90
170:IF O$<>"Y" THEN
    CURSOR 0,3: GOTO 140
180:GOSUB 430
190:GOTO 140
200:GOSUB 580
210:CLS
220:END
230:L=1
240:FOR I=0 TO N
250:FOR J=0 TO N
260:AA$="A("+ STR$ (I+1)
    +","+ STR$ (J+1)+")=
    "
270:PRINT AA$;
280:CURSOR LEN (AA$),L
290:INPUT A(I,J)
300:L=L+1: IF L=4 THEN
    LET L=3
310:CURSOR
320:NEXT J
330:NEXT I
340:RETURN
350:FOR I=0 TO N
360:FOR J=0 TO N
370:AA$="A("+ STR$ (I+1)
    +","+ STR$ (J+1)+")=
    "
380:LPRINT AA$;A(I,J)
390:NEXT J
400:NEXT I
410:LPRINT : LPRINT
420:RETURN
430:CLS
440:PRINT "X=          Y="
450:CURSOR 2,0: INPUT X
460:IF (X<1)+(X>N+1)
    THEN 450
470:CURSOR 10,0: INPUT Y
480:IF (Y<1)+(Y>N+1)
    THEN 470
490:CURSOR
500:AA$="A("+ STR$ (X)+"
    ,"+ STR$ (Y)+")="
510:PRINT AA$;A(X-1,Y-1)
520:PRINT AA$
530:CURSOR LEN (AA$),2
540:INPUT A(X-1,Y-1)
550:CURSOR
560:F=1
570:RETURN
580:FOR M=N TO 1 STEP -1
590:P=A(M,M)
600:IF P=0 THEN 760
610:FOR I=0 TO M-1
620:Q=A(I,M)/P
630:FOR J=0 TO M
640:A(I,J)=A(I,J)-Q*A(M,
    J)
650:NEXT J
660:NEXT I
670:NEXT M
680:D=A(0,0)
690:FOR I=1 TO N
700:D=D*A(I,I)
710:NEXT I
720:BEEP 1
730:LPRINT "det= ";D
740:LPRINT : LPRINT
750:RETURN
760:BEEP 2: CLS
770:PRINT "CALCULATE IMP
    OSSIBLE !!"
780:END
```

985 bytes

■ MEMORY CONTENTS

A	
B	
C	
D	matrix
E	
F	flag
G	
H	
I	count
J	count
K	
L	√
M	count
N	degrees
O$	flag
P	√
Q	√
R	
S	
T	
U	
V	
W	
X	change in matrix elements
Y	change in matrix elements
Z	
AA$	display
A(N, N)	elements of matrix

Program Title: MODIFIED MOVING AVERAGE

Thermal printer cassette interface CE-126P required

■ OVERVIEW

Even changes difficult to read can be grasped at a glance!
Modified moving average is used for when you search for the average point at all fixed parts in the data and further, to continue, when you want to know overall data trends. In concrete terms, it is possible to grasp accurately the long-term trends of such items as sales returns, the price of stocks and the rate of exchange which are subject to volatile changes from day to day.

■ INSTRUCTIONS

1. Program starts at [DEF] [A] . Input the number of items, and to continue, input further elements. By inputting each element the input data will be output to the printer, and also the input data and the average values will be output to the printer from n + 1st.

2. When the program has been completed, control is completed by operating only [ENTER] when it is waiting for data input.

■ CONTENTS

Processing varies with the number of averaging items (n) being an odd number or even number.

1. n is an odd number:

$$\bar{X}_1 = \sum_{i=1}^{n} Xi/n$$

$$\bar{X}_2 = \sum_{i=2}^{n+1} Xi/n$$

$$\vdots$$

2. n is an even number:

$$\bar{X}_1 = \left(\frac{X_1}{2} + \frac{X_{n+1}}{2} + \sum_{i=2}^{n} Xi\right)/n$$

$$\bar{X}_2 = \left(\frac{X_2}{2} + \frac{X_{n+2}}{2} + \sum_{i=3}^{n+1} Xi\right)/n$$

$$\vdots$$

■ EXAMPLE

Input the following data and search for the modified moving average. Set the number of items at 4.

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}
82	15	8	53	0	41	22	3	18	75	64	39	46

■ PRINTED OUTPUTS

```
 * X(1)=82.
 * X(2)=15.
 * X(3)=8.
 * X(4)=53.
 * X(5)=0.
  Average=29.25
 * X(6)=41.
  Average=22.25
 * X(7)=22.
  Average=27.25
 * X(8)=3.
  Average=22.75
 * X(9)=18.
  Average=18.75
 * X(10)=75.
  Average=25.25
 * X(11)=64.
  Average=34.75
 * X(12)=39.
  Average=44.5
 * X(13)=46.
  Average=52.5
```


■ KEY OPERATION SEQUENCE

1) DEF A [Program start]

```
NUMBER OF ITEMS=_
```

2) 4 ENTER [Number of items input]

```
NUMBER OF ITEMS=4
X(1)=?
```

3) 82 ENTER [X(1) input]

```
NUMBER OF ITEMS=4
X(1)=82
X(2)=?
```

.
.
.

input as above

.
.
.

4) 46 ENTER [X(13) input]

```
X(11)=64
X(12)=39
X(13)=46
X(14)=?
```

5) ENTER [Program completed]

```
>
```

■ PROGRAM LIST

```
 10:REM ***************
 20:REM MODIFIED MOVING
 25:REM     AVERAGE
 30:REM ***************
 40:"A": CLEAR : CLS :
    WAIT 0
 50:N=1:E=0:C=1
 60:INPUT "NUMBER OF ITE
    MS=";A
 70:DIM X(A-1)
 80:IF A<> INT (A/2)*2
    THEN 220
 90:FOR I=0 TO A-1
100:GOSUB 430
110:NEXT I
120:FOR I=0 TO A-1
130:PRINT "X("+ STR$ (N)
    +")=";
140:CURSOR 5+ LEN (
    STR$ (N)),C: INPUT D
    : PRINT : GOTO 160
150:LPRINT : LPRINT :
    CLS : END
160:E=E+D
170:LPRINT "* X("+ STR$
    (N)+")=";D
180:LPRINT " Average=";(
    E-.5*(D+X(I)))/A
190:E=E-X(I):X(I)=D:N=N+
    1:C=C+1: IF C=4 LET
    C=3
200:NEXT I
210:GOTO 120
220:FOR I=0 TO A-2
230:GOSUB 430
240:NEXT I
250:B=A-1
260:PRINT "X("+ STR$ (N)
    +")=";
270:CURSOR 5+ LEN (
    STR$ (N)),C
280:INPUT X(B): PRINT
290:LPRINT "* X("+ STR$
    (N)+")=";X(B)
300:LPRINT "  Average=";
    E/A
310:E=E+X(B):N=N+1:C=C+1
    : IF C=4 LET C=3
320:FOR I=0 TO B
330:PRINT "X("+ STR$ (N)
    +")="
340:CURSOR 5+ LEN (
    STR$ (N)),C
350:INPUT D: PRINT :
    GOTO 370
360:LPRINT : LPRINT :
    CLS : END
370:E=E-X(I)+D:X(I)=D
380:LPRINT "* X("+ STR$
    (N)+")=";D
390:LPRINT "  Average=";
    E/A
400:N=N+1:C=C+1: IF C=4
    LET C=3
410:NEXT I
420:GOTO 320
430:PRINT "X("+ STR$ (N)
    +")=";
440:CURSOR 5+ LEN (
    STR$ (N)),C
450:INPUT D: PRINT
460:LPRINT "* X("+ STR$
    (N)+")=";D
470:E=E+D:X(I)=D:N=N+1:C
    =C+1: IF C=4 THEN
    LET C=3
480:RETURN
```

821 bytes

■ MEMORY CONTENTS

A	number of items
B	√
C	√
D	data
E	data total
F	
G	
H	
I	loop counter
J	
K	
L	
M	
N	√
O	
P	
Q	
R	
S	
T	
U	
V	
W	
X	
Y	
Z	
X(A−1)	data

Program Title: SINGLE ARRANGEMENT METHOD

Thermal printer cassette interface CE-126P required

■ OVERVIEW

A help in fraction analysis.

This program will greatly help to relieve the troubles that accompany the number of levels, the number of replications and for each data element it will calculate the square sum (S), the degree of freedom (DF), the unbiased estimate of population variance (V) and the unbiased estimate of population variance ratio (F).

■ INSTRUCTIONS

1. Program starts when DEF A is pressed.
   - Select the output method of the calculation results. (Printer 1 ENTER , display 2 ENTER).
   - Input the number of levels and the number of replications. To continue, input the data according to the display.
   - When all data input has been completed, select whether to print out the input contents or not.
2. Next, select whether to amend the data entered or not.
   - When you amend the data, operate at Y ENTER , furthermore, input the number of rows and columns of the element to be amended and, to follow, input the correct data.
   - When there is no need to amend, operate at N ENTER .
3. Output the calculation results and complete control.
   - When data amendment has been carried out, select whether to print out input details once again, and operate N ENTER when prompted to amend the input data or not.

■ CONTENTS

No. of levels: a

No. of replications: n

Data: X_{ij} $(i = 1 \sim a,\ j = 1 \sim n)$ No. of data: an

1. $[X] = x^2/an$

   $[A] = x^2_i/n$

   $[AS] = \Sigma x^2_{ij}$

2. $S_A = [A] - [X]$

   $S_T = [AS] - [X]$

   $S_E = [AS] - [A]$

3. $\phi_A = a - 1$

   $\phi_E = a_n - a$

   $\phi_T = a_n - 1$

4. $\{V\} = \{S\} / \{\phi\}$

5. $F = V_A / V_E$

V_T is not calculated.

■ EXAMPLE

In the chart below, manure was given to plant shoots in 3 separate stages and the growth after 1 week was measured. From these results we are to look for square sums degree of freedom, unbiased variance and unbiased variance ratio.

		Growth after 1 week [cm]				
Manure [g/sold]	10	7.80	7.64	7.49	7.72	7.78
	20	8.02	8.28	8.16	8.31	8.36
	30	8.52	8.41	8.54	8.49	8.37

■ PRINTED OUTPUTS

```
Factor value=
                              3.
Replication value=
                              5.

X(1,1)= 7.8
X(1,2)= 7.64
X(1,3)= 7.49
X(1,4)= 7.72
X(1,5)= 7.78

X(2,1)= 8.02
X(2,2)= 8.28
X(2,3)= 8.16
X(2,4)= 8.31
X(2,5)= 8.36

X(3,1)= 8.52
X(3,2)= 8.41
X(3,3)= 8.54
X(3,4)= 8.49
X(3,5)= 8.37
```

```
*Square sums
a=1.596
e=0.15956
t=1.75556

*Degree of freedom
a=2.
e=12.
t=14.

*Unbiased variance
a=0.798
e=1.329666667E-02

*Unbiased variance ratio
a=60.01504135
```

■ KEY OPERATION SEQUENCE

1) DEF A [Program start]

```
*SINGLE ARRANGE. METHOD*

PRINTER ON=1   OFF=2_
```

2) 1 ENTER
[Output method select output to printer]

```
*SINGLE ARRANGE. METHOD*
NO. OF FACTORS=_
```

3) 3 ENTER 5 ENTER
[No. of factors and no. of replications input]

```
*SINGLE ARRANGE. METHOD*
NO. OF FACTORS=3
NO. OF REPLICATIONS=5
X(1,1)=?
```

4) 7.8 ENTER 7.64 ENTER
7.58 ENTER [Each data input]

```
X(1,1)=7.8
X(1,2)=7.64
X(1,3)=7.58
X(1,4)=?
```

.
.
.
input as above
.
.
.

5) 8.54 ENTER 8.49 ENTER
8.37 ENTER [Each data input]

```
X(3,3)=8.54
X(3,4)=8.49
X(3,5)=8.37
PRINT OUT DATA(Y/N)_
```

6) N ENTER

```
X(3,4)=8.49
X(3,5)=8.37
PRINT OUT DATA(Y/N)N
AMEND DATA(Y/N)_
```

7) Y ENTER [Amend data]

```
X=?          Y=
```

8) 1 ENTER
[Number of columns input]

```
X=1          Y=?
```

9) 3 ENTER
[Number of rows input]

```
X=1          Y=3
X(1,3)=7.58
X(1,3)=?
```

10) 7.49 ENTER
[Correct data input]

```
X=1          Y=3
X(1,3)=7.58
X(1,3)=7.49
AMEND DATA(Y/N)_
```

11) N ENTER

```
X(1,3)=7.58
X(1,3)=7.49
AMEND DATA(Y/N)N
PRINT OUT DATA(Y/N)_
```


12) Y ENTER
[Print out data list after amendment]

```
X(1,3)=7.49
AMEND DATA(Y/N)N
PRINT OUT DATA(Y/N)Y
AMEND DATA(Y/N)_
```

13) N ENTER
[Output results program completed]

```
>
```

■ PROGRAM LIST

```
 5:REM ***************
10:REM SINGLE ARRANGE.
20:REM       METHOD
30:REM ***************
40:"A": CLEAR : WAIT 0:
   CLS
50:BEEP 1
60:CURSOR 0,0: PRINT "*
   SINGLE ARRANGE. METH
   OD*"
70:CURSOR 2,2: INPUT "P
   RINTER ON=1 OFF=2 ";
   P
80:IF (P<>1)*(P<>2)
   THEN 70
90:CURSOR 0,2: PRINT "
      "
100:CURSOR 0,1: INPUT "N
   O. OF FACTORS=";A
110:CURSOR 0,2: INPUT "N
   O. OF REPLICATIONS="
   ;N
120:CURSOR
130:DIM D(A,N)
140:FOR I=1 TO A
150:FOR J=1 TO N
160:BB$="X("+ STR$ (I)+"
   ,"+ STR$ (J)+")= "
170:CURSOR : PRINT BB$
180:CURSOR LEN (BB$),3
190:INPUT D(I,J)
200:NEXT J
210:NEXT I
220:IF (P=2)*(F=0) THEN
   370
230:IF (P=2)*(F=1) THEN
   520
240:CURSOR :F=0: INPUT "
   PRINT OUT DATA(Y/N)"
   ;O$
250:IF O$="N" THEN 370
260:IF O$<>"Y" THEN 240
270:LPRINT : LPRINT "Fac
   tor value=": LPRINT
   A
280:LPRINT "Replication
   value=": LPRINT N:
   LPRINT
290:FOR I=1 TO A
300:FOR J=1 TO N
310:BB$="X("+ STR$ (I)+"
   ,"+ STR$ (J)+")= "
320:LPRINT BB$;D(I,J)
330:NEXT J
340:LPRINT
350:NEXT I
360:LPRINT
370:CURSOR : INPUT "AMEN
   D DATA(Y/N)";O$
380:IF (O$="N")*(F=0)
   THEN 520
390:IF (O$="N")*(F=1)
   THEN 220
400:IF O$<>"Y" THEN 370
410:CLS :F=1
420:PRINT "X=        Y="
430:CURSOR 3,0: INPUT X
440:IF (X<1)+(X>A) THEN
   430
450:CURSOR 13,0: INPUT Y
   : CURSOR
460:IF (Y<1)+(Y>N) THEN
   450
470:BB$="X("+ STR$ (X)+"
   ,"+ STR$ (Y)+")= "
480:PRINT BB$;D(X,Y)
490:PRINT BB$
500:CURSOR LEN (BB$),2:
   INPUT D(X,Y):
   CURSOR
510:GOTO 370
520:FOR I=1 TO A
530:E=0
540:FOR J=1 TO N
550:E=E+D(I,J)
560:Z=Z+D(I,J)*D(I,J)
570:NEXT J
580:S=S+E:T=T+E*E
590:NEXT I
600:S=S*S/(A*N):T=T/N:T=
   T-S:Z=Z-S:L=Z-T:F=A-
   1
610:U=T/F:O=A*(N-1):M=L/
   O:Q=A*N-1:G=U/M
620:ON P GOSUB 640,720
630:CLS : END
640:LPRINT "*Square sums
   "
650:LPRINT "a=";T:
   LPRINT "e=";L:
   LPRINT "t=";Z:
   LPRINT
660:LPRINT "*Degree of f
   reedom"
670:LPRINT "a=";F:
   LPRINT "e=";O:
   LPRINT "t=";Q:
   LPRINT
680:LPRINT "*Unbiased va
   riance"
690:LPRINT "a=";U:
   LPRINT "e=";M:
   LPRINT
700:LPRINT "*Unbiased va
   riance ratio"
710:LPRINT "a=";G:
   LPRINT : LPRINT :
   RETURN
720:CLS : PRINT "*Square
    sums"
730:PRINT "a=";T: PRINT
   "e=";L: WAIT :
   PRINT "t=";Z: WAIT
740:WAIT 0: CLS : PRINT
   "*Degree of freedom"
750:PRINT "a=";F: PRINT
   "e=";O: WAIT :
   PRINT "t=";Q
760:WAIT 0: CLS : PRINT
   "*Unbiased variance"
770:PRINT "a=";U: WAIT :
   PRINT "e=";M
780:WAIT 0: CLS : PRINT
   "*Unbiased variance
   ratio"
790:WAIT : PRINT "a=";G
800:CLS : RETURN
```

1533 bytes

■ MEMORY CONTENTS

A	no. of factors
B	
C	
D	
E	column data sum
F	flag degree of freedom
G	√
H	
I	loop counter
J	loop counter
K	
L	√
M	variance
N	no. of replications
O, O$	√, √
P	√
Q	√
R	
S	data sum
T	√
U	variance
V	
W	
X	amendment
Y	amendment
Z	√
BB$	display
D(A, N)	data

Program Title: F TEST

Thermal printer cassette interface CE-126P required

■ OVERVIEW

To help you carry out the F test smoothly.
The F test is very effective, when you want to know whether there is a relationship between conditions and results in experiments or not. With this program you can carry out troublesome, necessary calculations that you make when you do this test. When you input data numbers and standard deviations or data numbers and data, the F test value and the degree of freedom will be printed out.

■ INSTRUCTIONS

DEF [A] operation → when you know the standard deviation

1. Input the data numbers in group 1, standard deviation, the data numbers in group 2 and standard deviation, according to the display.
2. When input has been completed, the input details and F test value and the degree of freedom will be printed out and the program will be completed.

DEF [B] operation → when you do not know the standard deviation

1. Input the data numbers in group 1 and each element, and follow with the data numbers in group 2 and each element, according to the display.
2. When input has been completed, the input details and the average, standard deviation, the F test value and the degree of freedom will be printed out and the program will be completed.

■ CONTENTS

- Calculation of the sum of squares

  Sum of square between classes $= \Sigma n_i(\bar{X}_i - \bar{X})^2$

  Sum of square within class $= \Sigma\Sigma(X_{ij} - \bar{X}_i)^2$

  Total $= \Sigma\Sigma(X_{ij} - \bar{X})^2$

  $\bar{X}_i$: Average within group

  $\bar{X}$: Total average of data

- Calculation of degrees of freedom

  Degrees of freedom between classes = No. of groups − 1

  Degrees of freedom within class = Total of data − Degrees of freedom between classes − 1

  Total = Total of data − 1

- Calculation of square of average

  Square of average between classes = Sum of square between classes / Degrees of freedom between classes

  Square of average within class = Sum of square within class / Degrees of freedom within class

- Calculation of F (Variance ratio)

  F = Square of average between classes / Square of average within class

■ EXAMPLES

1. Input the following table and search for the F test value and the degree of freedom.

group 1	data numbers: 10
	standard deviation: 0.326114176

group 2	data numbers: 8
	standard deviation: 0.356452736

2. Input the following data and search for the average, standard deviation, F test value and the degree of freedom in each of the groups.

	1	2	3	4	5	6	7
group 1	70	67	82	89	93	76	81
group 2	110	108	101	98	99	96	103

■ PRINTED OUTPUTS

[Example 1]

```
******   F TEST   ******

Data numbers 1=
                      10.
Std. deviation 1=
              0.326114176
Data numbers 2=
                       8.
Std. deviation 2=
              0.356452736
------------------------

F Test value=
             1.194715641

Degree of freedom 1=
                       7.
Degree of freedom 2=
                       9.
```

[Example 2]

```
******   F TEST   ******

Data numbers 1=
                       7.
Data 1(1)=70.
Data 1(2)=67.
Data 1(3)=82.
Data 1(4)=89.
Data 1(5)=93.
Data 1(6)=76.
Data 1(7)=81.

Average   1=
              79.71428571
Std. deviation 1=
              9.481812206

Data numbers 2=
                       7.
Data 2(1)=110.
Data 2(2)=108.
Data 2(3)=101.
Data 2(4)=98.
Data 2(5)=99.
Data 2(6)=96.
Data 2(7)=103.

Average   2=
             102.1428571
Std. deviation 2=
              5.209881704

------------------------

F Test value=
              3.312279483

Degree of freedom 1=
                       6.
Degree of freedom 2=
                       6.
```


■ KEY OPERATION SEQUENCE

[In the case of example 1]

1) DEF A [Starting program]

```
DATA NUMBERS1=_
```

2) 10 ENTER
[Data numbers input from group 1]

```
DATA NUMBERS1=10
STD. DEVIATION1=_
```

3) .326114176 ENTER
[Standard deviation input from group 1]

```
DATA NUMBERS1=10
STD. DEVIATION1=.32114176
DATA NUMBERS2=_
```

4) 8 ENTER
[Data number input from group 2]

```
DATA NUMBERS1=10
STD. DEVIATION1=.326114176
DATA NUMBERS2=8
STD. DEVIATION2=_
```

5) .356452736 ENTER
[Standard deviation input from group 2]

```
STD.DEVIATION1=.326114176
DATA NUMBERS2=8
STD.DEVIATION2=.356452736
>
```

(Output results control completed)

[In the case of example 2]

1) DEF B [Starting program]

```
DATA NUMBERS1=_
```

2) 7 ENTER
[Data numbers input from group 1]

```
DATA NUMBERS1=7
DATA 1(1)=?
```

3) 70 ENTER
[Data 1 input from group 1]

```
DATA NUMBERS1=7

DATA 1(1)=70
DATA 1(2)=?
```

⋮
input as above
⋮

4) 81 ENTER
[Data 2 input from group 1]

```
DATA NUMBERS2=_
```

5) 7 ENTER
[Data numbers input from group 2]

```
DATA NUMBERS2=7

DATA 2(1)=?
```

⋮
input as above
⋮

6) 96 ENTER
[Data 6 input from group 2]

```
DATA 2(4)=98
DATA 2(5)=99
DATA 2(6)=96
DATA 2(7)=?
```

7) 103 ENTER
[Data 7 input from group 2]

```
DATA 2(5)=99
DATA 2(6)=96
DATA 2(7)=103
>
```

(Output results control completed)

■ PROGRAM LIST

```
 10:REM ****************
 20:REM *     F TEST     *
 30:REM ****************
 40:"A": CLEAR : CLS :
    WAIT 0
 50:INPUT "DATA NUMBERS
    1=";M
 60:INPUT "STD. DEVIATIO
   N 1=";R
 70:INPUT "DATA NUMBERS
    2=";N
 80:INPUT "STD. DEVIATIO
    N 2=";S
 90:GOTO 330
100:"B": CLEAR : CLS :
    WAIT 0:F=1
110:INPUT "DATA NUMBERS
    1=";M
120:DIM X(M)
130:PRINT ""
140:L=2
150:FOR I=1 TO M
160:CURSOR : PRINT "DATA
     1("+ STR$ (I)+")=";
170:CURSOR 9+ LEN (
    STR$ (I)),L: INPUT X
    (I)
180:X=X+X(I):R=R+X(I)*X(
    I)
190:L=L+1: IF L=4 LET L=
    3
200:NEXT I
210:X=X/M:R=R-M*X*X:R=
    SQR (R/(M-1))
220:CLS
230:INPUT "DATA NUMBERS
    2=";N
240:DIM Y(N)
250:PRINT "":L=2
260:FOR I=1 TO N
270:CURSOR : PRINT "DATA
     2("+ STR$ (I)+")=";
280:CURSOR 9+ LEN (
   STR$ (I)),L: INPUT Y
    (I)
290:Y=Y+Y(I):S=S+Y(I)*Y(
    I)
300:L=L+1: IF L=4 LET L=
    3
310:NEXT I
320:Y=Y/N:S=S-N*Y*Y:S=
    SQR (S/(N-1))
330:T=R*R:U=S*S:O=M:Q=N
340:IF U>T LET P=M:M=N:N
    =P:P=U:U=T:T=P
350:LPRINT "******   F T
    EST   ******"
360:LPRINT
370:LPRINT "Data numbers
     1=": LPRINT O
380:IF F<>1 THEN GOTO 44
    0
390:FOR I=1 TO O
400:LPRINT "Data 1("+
    STR$ (I)+")=";X(I)
410:NEXT I
420:LPRINT
430:LPRINT "Average    1=
   ": LPRINT X
440:LPRINT "Std. deviati
    on 1=": LPRINT R
450:LPRINT
460:LPRINT "Data numbers
     2=": LPRINT Q
470:IF F<>1 THEN GOTO 53
    0
480:FOR I=1 TO Q
490:LPRINT "Data 2("+
    STR$ (I)+")=";Y(I)
500:NEXT I
510:LPRINT
520:LPRINT "Average    2=
   ": LPRINT Y
530:LPRINT "Std. deviati
    on 2=": LPRINT S
540:LPRINT : LPRINT "---
    ------------------
    -": LPRINT
550:LPRINT "F Test value
    =": LPRINT T/U
560:LPRINT
570:LPRINT "Degree of fr
    eedom 1=": LPRINT M-
    1
580:LPRINT "Degree of fr
    eedom 2=": LPRINT N-
    1
590:LPRINT : LPRINT
600:END
```

1094 bytes

■ MEMORY CONTENTS

A	
B	
C	
D	
E	
F	√
G	
H	
I	loop counter
J	
K	
L	√
M	data numbers 1
N	data numbers 2
O	√
P	√
Q	√
R	standard variable 1
S	standard variable 2
T	variance 1
U	variance 2
V	
W	
X	data 1 sum, square
Y	data 2 sum, square
Z	
X(M)	data 1
Y(N)	data 2

INDEX

& 38
* 47
+ 47
− 47
/ 47
^ 47
√ 163
< 47
◀ 21
<= 47
<> 47
= 47
> 47
▶ 21
>= 47
π 159
↑ 198
↓ 198
ABS 160
ACS 160
ALL RESET 14
AND 48
AREAD 106
ASC 164
ASN 160
ATN 160
Arrays 42
Auto off (Auto Power Off) 18
BASIC 194
BEEP 108
Batteries 15
CA key 10
CE-126P 73
CHAIN 109
CHR$ 164
CLEAR 111
CLOAD 92
CLOAD? 94
CLOSE 179
CLS 112
Clear key 10
CONSOLE 180
CONT 94
COS 161
CSAVE 96
CURSOR 113
Cursor 12
Cassette 75
Character Code Chart 206
Commands 53,88
Constants 38
DATA 116
DEF key 69
DEG 161
DEGREE 117
DELete key 24
DIM 118
DMS 161
Debugging 198
Display 12
END 120
ENTER key 8,19
EXP 161
Editing calculations 21
Error Messages 203
Exponentiation 161
Expressions 46
FOR ...TO...STEP 121
Formatting output 209
Functions 51,91
GCURSOR 166
GOSUB 123
GOTO 97,124
GPRINT 168
GRAD 125
Graphic Functions 166
Hexadecimal 38
IF ... THEN 126
INKEY$ 156
INPUT 127
INPUT# 129
INPUT#1 181

INSert key 57
INT 161
LEFT$ 164
LEN 165
LET 132
LINE 173
LIST 98
LLIST 99,182
LN 161
LOAD 184
LOG 162
LPRINT 133,185
Labelled programs 69
Limits of numbers 31
Line numbers 52
Logical expressions 48
Loops 61
MEM 157
MEM "B" 157
MEM "C" 157
MEM$ 159
MERGE 100
MID$ 165
Maintenance 202
Masks 209
Memory Protection 62
NEW 103
NEXT 135
NOT 49
Numeric expressions 47
Numeric variables 40
ON(Start up) 18
ON...GOSUB 136
ON...GOTO 137
OPEN 188
OPEN$ 190
OR 50
Operator precedence 50
Operator priority 213
Operators 47
P↔NP 73
PASS 104
PAUSE 138
PI 159
PRINT 140
PRINT# 142
PRINT#1 191
PROgram mode 54
Parentheses 50
POINT 176
Power 18
PRESET 177
Printer 73
Priority 213
Program 52
Pseudovariables 156
PSET 178
RADIAN 144
RAM Card 81
RANDOM 145
READ 146
REM 147
RESET 14
RESTORE 148
RETURN 149
RIGHT$ 165
RND 162
RUN 105
RUN mode 51,54
Range of numbers 31
ReSerVe mode 70
Relational expression 47
SAVE 193
SGN 162
SHIFT key 19
SIN 162
SMaLl key 12
SQR 163
STOP 150
STR$ 165
Scientific notation 30
Serial I/O Functions 179
Square root 163
Statements 52
String expressions 47
String variables 40
Subroutines 123
TAN 163

TEXT	195
TROFF	151
TRON	152
Templates	72
Troubleshooting	197
USING	153
VAL	165
Variables	38
Verbs	106
WAIT	155

SERVICE CENTER ADDRESS

SHARP ELECTRONICS CORPORATION
9 Sharp Plaza, Paramus, New Jersey 07652
(201) 265-5600

SHARP ELECTRONICS CORPORATION
6478 Interstate 85 Norcross, Georgia 30093
(404) 448-5230

SHARP ELECTRONICS CORPORATION
430 East Plainfield Road, Countryside, Illinois 60525
(312) 482-9292

SHARP ELECTRONICS CORPORATION
Sharp Plaza,
20600 South Alameda St., Carson, California 90810
(213) 637-9488

SHARP ELECTRONICS CORPORATION
1205 Executive Drive, East Richardson, Texas. 75081
(214) 234-1136

SHARP ELECTRONICS CORPORATION

CORPORATE HEADQUATERS AND EXECUTIVE OFFICES:
10 Sharp Plaza, Paramus, New Jersey 07652, Phone: (201) 265-5600

REGIONAL SALES OFFICES AND DISTRIBUTION CENTERS:
Eastern: 10 Sharp Plaza, Paramus, New Jersey 07652, Phone: (201) 265-5600
Midwest: 430 East Plainfield Road CountrySide, Illinois 60525, Phone: (312) 482-9292
Western: 20600 South Alameda Street, Carson, California 90810, Phone: (213) 637-9488

© 1984 SHARP CORPORATION
Printed in Japan
5C0.9T(TINSE4218CCZZ)④